Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-024-02524-x

A systematic online update method for reduced-order-model-based digital twin

Yifan Tang[1] · Pouyan Sajadi[1] · Mostafa Rahmani Dehaghani[1] · G. Gary Wang[1]

Received: 16 May 2024 / Accepted: 5 November 2024

Abstract
A digital twin (DT) is a model that mirrors a physical system and is continuously updated with real-time data from the physical system. Recent implementations of reduced-order-model-based DT (DT-ROM) have been applied in aerodynamics and structural health monitoring, where partial differential equations (PDEs) are utilized to update reduced bases and coefficients. However, these methods are not directly applicable when the PDEs of the system are unknown. This paper addresses the online update challenge for DT-ROM in scenarios lacking known PDEs of the system. To tackle the challenge, a systematic online update and application method is proposed. During the online update, the projection residual of online data on the reduced bases determines the necessity of updating reduced bases; the prediction residual of online data obtained by the current DT-ROM is used to decide whether to update the coefficient model. By sequentially evaluating both criteria, the method selectively incorporates essential online data for the online DT model update. During the online application, a criterion defined based on online data is adopted to determine whether the offline DT-ROM or the online one is applied to output final predictions. The capability of the proposed method is tested through three numerical and three engineering problems. Results indicate that the proposed online update method consistently reduces both projection and prediction residuals, thereby progressively enhancing the performance of the online DT-ROM on test data. Meanwhile, the online application method provides a prediction performance better than using offline DT-ROM only. Both demonstrate that the proposed work could be applied to online DT updates where the PDEs of the system are unknown.

Keywords Digital twin · Reduced-order model · Online update and application · Partial differential equation · Twin update

Introduction

Literature review

The digital twin (DT) was first conceptualized in the 1960s when NASA developed a ground-based simulator to forecast the conditions of Apollo spacecraft in space. This terminology was later formally introduced by Michael Grieves at the University of Michigan in 2003 (Grieves, 2014). Subsequently, research and applications of DT have increased significantly in both academia and industry. Although the precise definition of DT varies depending on the context, it is generally recognized as a digital representation (or model) that simulates a physical system and possesses the capability to autonomously update itself based on data acquired from the corresponding physical system (Liu et al., 2021). Owing to its capacity for interaction with the physical realm, DT has been applied to various engineering scenarios (e.g., manufacturing (Xiang et al., 2019), process industry (Zhu & Ji, 2023) to support collaboration, information access, and decision-making (Altair Engineering, 2023). More detailed DT applications can be found in literature reviews (Lim et al., 2020; Su et al., 2023).

✉ Yifan Tang
yta88@sfu.ca

Pouyan Sajadi
sps11@sfu.ca

Mostafa Rahmani Dehaghani
mra91@sfu.ca

G. Gary Wang
gary_wang@sfu.ca

1 Product Design and Optimization Laboratory, School of Mechatronic Systems Engineering, Simon Fraser University, Surrey, BC, Canada

To facilitate effective DT applications across diverse tasks and conditions, the offline DT model construction and the online DT model update are essential prerequisites. For a specific physical system, the offline DT model construction involves designing a model based on prior knowledge, training this model with data collected from simulations or experiments, and refining the model to accurately reflect the system's responses or dynamics under a variety of conditions. The selection of DT models varies according to applications, incorporating methodologies such as partial differential equations (PDEs), finite element analysis (FEA), surrogate models (e.g., radial basis function (RBF), Gaussian process regression, or Kriging (KRG)), or machine learning models (e.g., artificial neural networks, support vector machines, or extreme learning machine (ELM)). Given that these conventional modeling approaches have been extensively studied for decades, this paper does not delve into the modeling methods of offline DT models.

When deploying offline DT models in certain scenarios, such as online control or monitoring, their prediction capabilities may be inadequate to provide acceptable online predictions. This shortcoming primarily arises due to the increased complexities or uncertainties inherent in online applications that are often simplified or disregarded during the offline modeling process. To address this issue, online update methods have been employed to update the offline DT model by integrating data acquired in real-time, thereby enhancing the DT's prediction accuracy.

Generally, online DT update methods could be classified according to the types of DT models involved. For DT models based on PDEs or FEA, online updates are designed to modify system parameters that have been estimated in offline models, such as mass or stiffness matrices. This process is commonly regarded as a parameter estimation problem, which has been addressed by Kalman filters (Qian et al., 2022), Particle filters (Eftekhar Azam & Mariani, 2018), and Bayesian estimation (Yuen & Kuok, 2011) and their various derivatives. Associated research areas include data assimilation (Cheng et al., 2023) and hybrid simulation (Al-Subaihawi et al., 2022). Although these online update methods have been applied successfully in fields such as structure health monitoring, agriculture, and fluid dynamics, the update formulas are designed based on the system's PDEs, rendering them inapplicable to scenarios where the PDEs are unknown.

When constructing offline DT models using data-driven methods such as surrogate models or machine learning models, online update methods serve two purposes, including the update of model parameters through optimization, Bayesian estimation or incremental computation (Huang et al., 2005), and the adaptive modification of model structures using Bayesian techniques (Yu et al., 2021) or heuristic strategies (Han et al., 2022). Areas relevant to these online update mechanisms encompass online learning (Hoi et al., 2021), incremental learning (Sarwar et al., 2020), lifelong learning (Parisi et al., 2019), and dynamic neural networks (Han et al., 2022). Nevertheless, these methods encounter the issue of catastrophic forgetting, where the knowledge acquired from offline data diminishes gradually during online updates, eventually disappearing. Consequently, the computation resource required for online updates increases as some of the online data are used to reinstate the knowledge that was initially captured by the offline data.

This paper examines offline DT models that are constructed as reduced-order models (ROM). Compared with PDEs and data-driven methods, ROM methods reduce the dimensionality and complexity of modeling while concurrently preserving predictive accuracy. In ROM, latent features of the system are identified from offline data through the application of proper orthogonal decomposition (POD), also known as principal component analysis and singular value decomposition. Mathematically, these latent features are the POD bases derived from the offline data. The system response is then approximated as a combination of these latent features. In the literature, two relevant online update methods are highlighted, i.e., online POD and online update of reduced-order PDEs.

(1) Online POD

In contrast to conventional methods where all data is accessible for POD computation, online POD is proposed to update POD components using limited data in real-time. Generally, online POD methods consume less computational resources and memory, making them suitable for integration into online applications. Based on computation strategies, online POD methods are categorized into moving (sliding) window POD (Badeau et al., 2004), incremental POD (Brand, 2002), and recurrent POD with matrix perturbation (Elshenawy et al., 2010). A comprehensive comparison of these methods is in Ref. (Cardot & Degras, 2018). Unlike online DT model update which aims to enhance the overall prediction performance of the system, online POD specifically improves the accuracy of the updated features, namely the POD bases.

(2) Online update of reduced-order PDEs

To improve the computational efficiency of high-dimensional PDEs, ROM has been employed to construct reduced-order PDEs for structural and fluid dynamics applications (Lu et al., 2021). These reduced-order PDEs can be updated online. For example, by fixing POD bases of mass and stiffness matrices in the motion equations, the corresponding coefficients of these bases are updated by employing the first-order optimality condition on an error estimator (Dorosti, 2017).

Additionally, in several aerodynamic applications, the updating of base coefficients is treated as an optimization problem solved by the Levenberg–Marquardt optimization method (Garbo & Bekemeyer, 2022; Mifsud et al., 2015) or the SNOPT toolbox (Vetrano et al., 2015). Further, to update the POD bases in reduced-order PDEs, a closed-form quadratic equation is formulated to determine the optimal scalar value in the updating formula, where the POD bases are revised by adding a scaled prior matrix (Griffiths et al., 2018). In certain structural health monitoring tasks, the POD bases and base coefficients are simultaneously updated, where the update is formulated as a dynamic model, solved using various Kalman filter methods (Ebrahimzadeh Hassanabadi et al., 2020, 2023; Eftekhar Azam & Mariani, 2018; Eftekhar Azam et al., 2017). While these applications illustrate the capability to update both POD bases and coefficients in engineering tasks online, a significant limitation is that they are predicated on the known PDEs of the systems. Consequently, these methods are not applicable if the system's PDEs are unknown.

The existing work that is most relevant to the online update task proposed in the paper was found in (Zhu & Ji, 2023), where the combination of information entropy, information gain ratio, similarity degree, and redundancy reduction was proposed to determine whether to update the POD bases with the online snapshot. However, they only focused on the reconstruction accuracy at known times/points, and a matrix with a fixed size was adopted for base coefficients instead of using a matrix as a function of times/points during the reconstruction. Therefore, their work cannot be directly applied to predict responses for unseen times or points.

Motivation

Diverging from the previously discussed online update tasks for ROM, this paper investigates a specific task of online DT update, as depicted in Fig. 1. This task involves a system whose dynamic model, such as the PDEs, remains undefined. The high-dimensional responses of some points are initially estimated using a simplified black-box (simulation) model, from which offline data are gathered. An offline DT model is then constructed based on the collected offline data to estimate the responses of the entire system. If actual responses of these points are sequentially measured during a physical experiment/process conducted under conditions identical to that used for offline modeling, this paper proposes utilizing the online data to update the offline DT model, thereby enhancing its predictive accuracy for the unseen (new) points in the system. Given that the experimental conditions for both offline and online data are consistent, the input variables, such as point location and experimental settings, remain constant for each point. This specific task has practical applications. For instance, a machine operator has obtained the offline DT model for a manufacturing process and wants to use it to offer operational guidance during the process. A maintenance team has the offline DT model for critical equipment or infrastructure and wants to regularly monitor the health status based on the DT model. Assuming the offline DT models in both applications are constructed according to the task definition, they are required to be updated based on online data from several sensors to maintain the prediction capability.

In engineering applications, the online data obtained from sensors or other equipment would be processed and cleaned before utilization, as the contained noises significantly affect the final application performance. In this paper, the preprocessing methods for online sensor data are not discussed, and the online data generated in the test cases (Sect. "Numerical problems" and Sect. "Additive manufacturing application") are assumed clean to be applied in online updates. For sensor data processing, the readers can refer to the review (Krishnamurthi et al., 2020).

This work aims to address the above needs. Contributions of this paper are listed below.

- A novel online DT update task is defined for systems without known PDEs. According to the authors' knowledge, this specific online update task is rarely or never discussed in the literature.
- An incremental POD with forgetting factor is proposed to efficiently capture features from online data. Different from conventional incremental POD methods which append new data to the end of the snapshot matrix, the proposed method allows the substitution of an existing snapshot vector with a new one without expanding the size of the snapshot matrix.
- A systematic online update and application method is proposed for the offline ROM model to update POD components and basis coefficient models sequentially. Different from current online update methods for ROM in DT applications, the proposed method does not rely on the PDEs of the system.

The remainder of the paper is structured as follows. Sect. "Methods" provides a review of ROM using POD and the conventional incremental POD method, based on which details of the proposed online update and application method are discussed. Sects. "Numerical problems" and "Additive manufacturing application" present testing results on three numerical problems and three engineering problems, respectively. Sect. "Discussion and future work" examines the impact of various hyperparameters in the proposed method. Finally, the summary of this paper is drawn in Sect. "Summary".

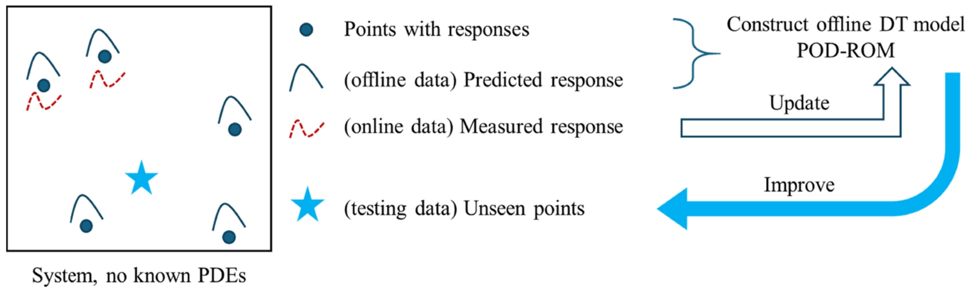

System, no known PDEs
Points with responses
(offline data) Predicted response
(online data) Measured response
(testing data) Unseen points
Construct offline DT model POD-ROM
Update
Improve

Fig. 1 Illustration of the online DT update task

Methods

Reduced order model with proper orthogonal decomposition

A reduced order model (ROM) represents a high-dimensional system as a linear combination of features (i.e., reduced bases) and coefficients, which are extracted from the system response matrix by POD. The extracted features are then assumed to be fixed during the reconstruction process, and a regression model (e.g., RBF, KRG, ELM, etc.) is selected to correlate coefficients with inputs of interest (e.g., time and boundary conditions). The high-dimensional system response of a new input is calculated based on the fixed features and the predicted coefficients. As the number of coefficients is much smaller than that of the system responses, the modeling complexity of the ROM method is lower than predicting the system responses from the inputs directly. In theory, the POD-based ROM model aims to find the minimum number of POD bases to reconstruct the system as accurately as possible, whose detailed process is presented below.

Considering a system whose input and response vectors are denoted as $\boldsymbol{x}_i \in R^{1\times n_{in}}$ and $\boldsymbol{y}_i \in R^{n_{out}\times 1}$ respectively, the input matrix and the snapshot matrix consisting of N samples are defined as $\boldsymbol{X} = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_N] \in R^{N\times n_{in}}$ and $\boldsymbol{Y} = \left[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\right] \in R^{n_{out}\times N}$. n_{in} is the input dimension. n_{out} is the dimension of the response vector with the condition $n_{out} \gg N$. The snapshot matrix is then decomposed by POD as:

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \left[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n_{out}}\right]\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & \lambda_N \\ & 0 & \end{bmatrix}\left[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N\right]^T \tag{1}$$

where $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n_{out}} \in R^{n_{out}\times 1}$ are left singular vectors (i.e., POD bases, or reduced bases); $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N \in R^{N\times 1}$ are right singular vectors; and $\lambda_1, \ldots, \lambda_N$ are singular values satisfying $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N \geq 0$. As 0 is a zero matrix with a size $(n_{out} - N) \times N$, Eq. (1) could be simplified as follows.

$$\boldsymbol{Y} = \boldsymbol{U}_N\boldsymbol{\Sigma}_N\boldsymbol{V}_N^T = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N]\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & \lambda_N \end{bmatrix}[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N]^T \tag{2}$$

To find the minimum number k ($k \leq N \ll n_{out}$, also known as the low rank) of POD bases and maintain the reconstruction accuracy, the energy percentage threshold ε is adopted as shown in Eq. (3) (Lu et al., 2021),

$$\frac{E_{cap}}{E_{total}} = \frac{\sum_{i=1}^{k} \lambda_i^2}{\sum_{i=1}^{N} \lambda_i^2} \geq \varepsilon \tag{3}$$

where $E_{total} = \sum_{i=1}^{N} \lambda_i^2$ is the total energy of the snapshot matrix, and E_{cap} is the energy captured by the former k POD bases. When finding the optimal number k, Eq. (2) is then rewritten as,

$$\begin{aligned}\boldsymbol{Y} &= \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & \lambda_k \end{bmatrix}[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k]^T \\ &= [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]\boldsymbol{C}_k \end{aligned}\tag{4}$$

where $\boldsymbol{C}_k = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N] = \boldsymbol{\Sigma}_k\boldsymbol{V}_k^T = [\lambda_1\boldsymbol{v}_1^T; \ldots; \lambda_k\boldsymbol{v}_k^T]$ is the coefficient matrix and $\boldsymbol{c}_i = \left[\lambda_1 v_1^i; \ldots; \lambda_k v_k^i\right] \in R^{k\times 1}$ is the coefficient vector for $\boldsymbol{y}_i$. The reconstruction formulation of $\boldsymbol{y}_i$ is then defined as Eq. (5),

$$\boldsymbol{y}_i = \boldsymbol{U}_k\boldsymbol{c}_i = \boldsymbol{U}_k[\lambda_1 v_1^i; \ldots; \lambda_k v_k^i] \tag{5}$$

where $v_1^i, \ldots, v_k^i$ are the i-th elements in the right singular vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ respectively. Meanwhile, the total energy is the sum of the squares of the magnitude of each snapshot vector (or coefficient vector) in mathematics,

$$E_{total} = \sum_{i=1}^{N} \|\boldsymbol{y}_i\|^2 = \sum_{i=1}^{N} \|\boldsymbol{c}_i\|^2 \tag{6}$$

where $\|\cdot\|$ is the Frobenius norm.

Based on the collected data $\{\boldsymbol{x}_i, \boldsymbol{c}_i | i \in [1, N]\}$, a coefficient model f is constructed as $\boldsymbol{c}_i = f(\boldsymbol{x}_i)$ to predict the coefficient vector based on the input. The final reconstructed response vector of a testing input $\boldsymbol{x}_{test}$ could be obtained directly as $\boldsymbol{y}_{test} = \boldsymbol{U}_k \boldsymbol{c}_{test} = \boldsymbol{U}_k f(\boldsymbol{x}_{test})$. For simplification, the ROM model using POD for feature extraction is denoted as POD-ROM, which is selected as the DT model in this paper.

Conventional incremental POD

In most applications, all high-dimensional snapshots are assumed to be accessible before computation. However, the offline POD based on all snapshots requires significant computation resources. To alleviate the computation burden, the incremental POD (IPOD) (Brand, 2002) was proposed to perform the POD computation based on an online snapshot, resulting in increased computational efficiency and robustness (Fareed et al., 2018; Li et al., 2022).

Given the snapshot matrix $\boldsymbol{Y} \in R^{n_{out} \times N}$ and the corresponding input matrix $\boldsymbol{X} \in R^{N \times n_{in}}$, the truncated POD is represented as $\boldsymbol{Y} = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T$ with a low-rank k. The incremental POD then updates $\boldsymbol{U}_k$, $\boldsymbol{\Sigma}_k$, and $\boldsymbol{V}_k$ with the online snapshot $\boldsymbol{y}' \in R^{n_{out} \times 1}$ based on the following equation, to capture the contained online information while maintaining the offline information:

$$\begin{aligned}\left[\boldsymbol{Y},\ \boldsymbol{y}'\right] &= \left[\boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T,\ \Delta \boldsymbol{y}' + \boldsymbol{U}_k \boldsymbol{U}_k^T \boldsymbol{y}'\right] \\ &= \left[\boldsymbol{U}_k \boldsymbol{j}\right] \begin{bmatrix} \boldsymbol{\Sigma}_k & \boldsymbol{U}_k^T \boldsymbol{y}' \\ 0 & p \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_k & 0 \\ 0 & 1 \end{bmatrix}^T \\ &= \boldsymbol{U}_{k+1} \boldsymbol{\Sigma}_{k+1} \boldsymbol{V}_{k+1}^T \end{aligned} \tag{7}$$

where $\Delta \boldsymbol{y}' = \boldsymbol{y}' - \boldsymbol{U}_k \boldsymbol{U}_k^T \boldsymbol{y}'$ is the projection residual of $\boldsymbol{y}'$ on the existing POD bases $\boldsymbol{U}_k$; $p = \|\Delta \boldsymbol{y}'\|$ and $\boldsymbol{j} = \Delta \boldsymbol{y}'/p$ are the corresponding magnitude and unit direction respectively. By performing the POD computation on the matrix $\boldsymbol{Q} \in R^{(k+1)\times(k+1)}$ with a size $k+1 \ll n_{out}$,

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{\Sigma}_k & \boldsymbol{U}_k^T \boldsymbol{y}' \\ 0 & p \end{bmatrix} = \boldsymbol{U}_Q \boldsymbol{\Sigma}_Q \boldsymbol{V}_Q^T \tag{8}$$

all POD components (i.e., left singular vectors, singular values, and right singular vectors) could be effectively updated online as Eq. (9).

$$\boldsymbol{U}_{k+1} = [\boldsymbol{U}_k \boldsymbol{j}] \boldsymbol{U}_Q,\ \boldsymbol{\Sigma}_{k+1} = \boldsymbol{\Sigma}_Q,\ \boldsymbol{V}_{k+1} = \begin{bmatrix} \boldsymbol{V}_k & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{V}_Q \tag{9}$$

According to the above equation, the newly added POD basis $\boldsymbol{U}_{k+1}[:, end]$ is a linear combination of $\boldsymbol{U}_k$ and $\boldsymbol{j}$. Considering that snapshots in $\boldsymbol{Y}$ are located in the space spanned by $\boldsymbol{U}_k$ and $\boldsymbol{U}_k$ are vertical to $\boldsymbol{j}$, the analytical projections of $\boldsymbol{Y}$ on $\boldsymbol{j}$ are all zero. As the unit direction $\boldsymbol{j}$ has the largest contribution in the newly added POD basis $\boldsymbol{U}_{k+1}[:, end]$, the projections of $\boldsymbol{Y}$ on $\boldsymbol{U}_{k+1}[:, end]$ are close to zero in most cases.

Generally, the number of POD bases increases with online snapshots adopted in IPOD, which requires more storage memory. To control the number of POD bases, the normalized projection residual threshold σ_{pro}^{IPOD} is used to determine whether the projection residual is neglectable during the update; meanwhile, the singular value threshold σ_{sv}^{IPOD} is adopted to determine whether to omit the newly added POD basis and its corresponding singular value from the updated components (Fareed et al., 2018; Li et al., 2022). More details are summarized in Table 1. Instead of the projection residual threshold applied in the conventional work (Fareed et al., 2018; Li et al., 2022), the normalized one is applied in the paper to reduce the effects of various magnitudes of online snapshots.

Table 1 Incremental POD (Adapted from (Fareed et al., 2018))

Input: $\boldsymbol{U}_k \in R^{n_{out}\times k}$, $\boldsymbol{\Sigma}_k \in R^{k\times k}$, $\boldsymbol{V}_k \in R^{N\times k}$, low rank k, online snapshot $\boldsymbol{y}' \in R^{n_{out}\times 1}$, normalized projection residual threshold σ_{pro}^{IPOD}, singular value threshold σ_{sv}^{IPOD}
Output: the updated POD components $\boldsymbol{U}_k$, $\boldsymbol{\Sigma}_k$, $\boldsymbol{V}_k$

```
1. $\boldsymbol{d} = \boldsymbol{U}_k^T \boldsymbol{y}'$, $\Delta \boldsymbol{y}' = \boldsymbol{y}' - \boldsymbol{U}_k \boldsymbol{d}$, $p = \|\Delta \boldsymbol{y}'\|$, $\boldsymbol{j} = \Delta \boldsymbol{y}'/p$
2. **If** $p/\|\boldsymbol{y}'\| > \sigma_{pro}^{IPOD}$
3. $\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{\Sigma}_k & \boldsymbol{d} \\ 0 & p \end{bmatrix} \in R^{(k+1)\times(k+1)}$
4. **Else**
5. $\boldsymbol{Q} = [\ \boldsymbol{\Sigma}_k\ \ \boldsymbol{d}\ ] \in R^{k\times(k+1)}$
6. **End**
7. Perform SVD: $\boldsymbol{Q} = \boldsymbol{U}_Q \boldsymbol{\Sigma}_Q \boldsymbol{V}_Q^T$
8. **If** $p/\|\boldsymbol{y}'\| > \sigma_{pro}^{IPOD}$
9. $\boldsymbol{U}_{new} = [\boldsymbol{U}_k \boldsymbol{j}] \boldsymbol{U}_Q$, $\boldsymbol{\Sigma}_{new} = \boldsymbol{\Sigma}_Q$, $\boldsymbol{V}_{new} = \begin{bmatrix} \boldsymbol{V}_k & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{V}_Q$
10. $k = k + 1$
11. **Elseif** $p/\|\boldsymbol{y}'\| < \sigma_{pro}^{IPOD}$ **or** $k \geq n_{out}$
12. $\boldsymbol{U}_{new} = \boldsymbol{U}_k \boldsymbol{U}_Q$, $\boldsymbol{\Sigma}_{new} = \boldsymbol{\Sigma}_Q$, $\boldsymbol{V}_{new} = \begin{bmatrix} \boldsymbol{V}_k & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{V}_Q$
13. **End**
14. **If** $\boldsymbol{\Sigma}_{new(k,k)} < \sigma_{sv}^{IPOD}$ and $\boldsymbol{\Sigma}_{new(k-1,k-1)} < \sigma_{sv}^{IPOD}$
15. $k = k - 1$
16. $\boldsymbol{U}_{new} = \boldsymbol{U}_{new(:,1:k)}$, $\boldsymbol{\Sigma}_{new} = \boldsymbol{\Sigma}_{new(1:k,1:k)}$, $\boldsymbol{V}_{new} = \boldsymbol{V}_{new(:,1:k)}$
17. **End**
18. $\boldsymbol{U}_k = \boldsymbol{U}_{new}$, $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_{new}$, $\boldsymbol{V}_k = \boldsymbol{V}_{new}$
```

Incremental POD with forgetting factor

When the number N of offline snapshots is large, the contribution of one online snapshot $\boldsymbol{y}'$ would be small or even neglectable in the conventional IPOD, where POD components have no significant updates. To alleviate the issue, the forgetting factor is adopted to balance the contribution of online and offline snapshots during the update process. The benefits of a forgetting factor have been demonstrated in works (Cardot & Degras, 2018; Ebrahimzadeh Hassanabadi et al., 2020; Li et al., 2000; Martinez-Ruiz & Lauro, 2023; Ross et al., 2008). When applying the conventional IPOD method, the forgetting factor $\beta \in [0,1]$ is generally integrated into the matrix $\boldsymbol{Q}$,

$$\boldsymbol{Q} = \begin{bmatrix} \beta\boldsymbol{\Sigma}_k & \boldsymbol{d} \\ 0 & p \end{bmatrix} \tag{10}$$

where $\beta = 0$ indicates forgetting all offline snapshots and only considering the current online snapshot during the update; $\beta = 1$ refers to no forgetting.

Although this strategy improves the contribution of online snapshots gradually, one limitation is that the size of the right singular vector matrix $\boldsymbol{V}_k$ increases from $N \times k_{old}$ to $(N + M) \times k_{new}$. k_{old} and k_{new} are the numbers of POD bases before and after the update with M online snapshots. If M is large, the required memory would be unacceptable. Meanwhile, storing information of all offline snapshots in the POD components seems unnecessary from the DT point. If the online snapshot of the input $\boldsymbol{x}_i (i \in [1, N])$ is accessible, the corresponding offline snapshot could be removed from $\boldsymbol{Y}$ to reduce the influence of the less accurate offline snapshot (than the corresponding online snapshot) for further DT updates.

This paper, therefore, proposes an incremental POD with a forgetting factor (IPOD-FF), where the offline snapshot $\boldsymbol{y}_i$ at $\boldsymbol{x}_i$ is allowed to be replaced with the corresponding online snapshot $\boldsymbol{y}'_i$. POD components are then updated accordingly. The detailed algorithm is summarized in Table 2 with the following descriptions.

When the offline snapshot $\boldsymbol{y}_i$ is determined to be removed, the total energy E_{total} is first reduced to E'_{total}, as shown in Eq. (11),

$$E'_{total} = E_{total} - \|\boldsymbol{y}_i\|^2 = E_{total} - \|\boldsymbol{c}_i\|^2 = E_{total} - \sum_{j=1}^{k} \left(\lambda_j v_j^i\right)^2 \tag{11}$$

where λ_j is the j-th singular value and v_j^i is the i-th element of the j-th right singular vector $\boldsymbol{v}_j$. The forgetting factor β is then calculated based on the total energy and

Table 2 Incremental POD with forgetting factor (IPOD-FF)

```
Input: $U_k \in R^{n_{out}\times k}$, $\Sigma_k \in R^{k\times k}$, $V_k \in R^{N\times k}$, low rank $k$, online snapshot $y'_i \in R^{n_{out}\times 1}$, normalized projection residual threshold $\sigma_{pro}^{IPOD}$, singular value threshold $\sigma_{sv}^{IPOD}$, total energy $E_{total}$
Output: updated $U_k$, $\Sigma_k$, $V_k$, $E_{total}$

1.Reduced energy $E'_{total} = E_{total} - \sum_{j=1}^{k}\left(\lambda_j v_j^i\right)^2$
2.Forgetting factor $\beta = \sqrt{\frac{E'_{total}}{E_{total}}}$
3.Remove elements $v_1^i, \ldots, v_k^i$ from $V_k$ to get the truncated $V_k^{-i}$ as shown in Eq. (13)
4.$\Sigma_k = \beta\Sigma_k$
5.$V_k^{-i} = V_k^{-i}/\beta$
6.Update $U_k$, $\Sigma_k$, $V_k$ with IPOD($U_k$, $\Sigma_k$, $V_k^{-i}$, $k$, $y'_i$, $\sigma_{pro}^{IPOD}$, $\sigma_{sv}^{IPOD}$) in Table 1
7.$E_{total} = E'_{total} + |y'_i|^2$
```

the reduced energy. Instead of selecting β value by trial-and-error or heuristics in conventional incremental POD methods (Ebrahimzadeh Hassanabadi et al., 2020; Martinez-Ruiz & Lauro, 2023), a deterministic calculation method is proposed in this work.

$$\beta = \sqrt{\frac{E'_{total}}{E_{total}}} \tag{12}$$

Meanwhile, the POD representation of the truncated snapshot matrix $\boldsymbol{Y}_{trunc}$, defined as the snapshot matrix excluding $\boldsymbol{y}_i$, is shown in Eq. (13).

$$\begin{aligned}\boldsymbol{Y}_{trunc} &= \left[\boldsymbol{y}_1, \ldots, y_{i-1}, y_{i+1}, \ldots, \boldsymbol{y}_N\right] \\ &= \boldsymbol{U}_k\left[\boldsymbol{c}_1, .., \boldsymbol{c}_{i-1}, \boldsymbol{c}_{i+1}, \ldots, \boldsymbol{c}_N\right] \\ &= \boldsymbol{U}_k\boldsymbol{\Sigma}_k\left(\boldsymbol{V}_k^{-i}\right)^T = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\begin{bmatrix} v_1^1 & \ldots & v_k^1 \\ \vdots & \ddots & \vdots \\ v_1^{i-1} & \ldots & v_k^{i-1} \\ v_1^{i+1} & \ldots & v_k^{i+1} \\ \vdots & \ddots & \vdots \\ v_1^N & \ldots & v_k^N \end{bmatrix}^T\end{aligned} \tag{13}$$

The forgetting factor is then applied to both $\boldsymbol{\Sigma}_k$ and $\boldsymbol{V}_k^{-i}$ as shown in Eq. (14), to maintain that the bases coefficients of snapshots could be reconstructed accurately from $\boldsymbol{\Sigma}_k$ and $\boldsymbol{V}_k^{-i}$ after performing the update. This would omit the requirement to store snapshots online.

$$\boldsymbol{\Sigma}_k = \beta\boldsymbol{\Sigma}_k \in R^{k\times k},\ \boldsymbol{V}_k^{-i} = \boldsymbol{V}_k^{-i}/\beta \in R^{(N-1)\times k} \tag{14}$$

Given the online snapshot $\boldsymbol{y}_i'$, the POD components could be updated via the conventional IPOD method directly. The snapshot matrix $\boldsymbol{Y}$ and the total energy E_{total} are then updated accordingly.

$$\boldsymbol{Y} = \left[\boldsymbol{Y}_{trunc},\ \boldsymbol{y}_i'\right],\ E_{total} = E_{total}' + \|\boldsymbol{y}_i'\|^2 \tag{15}$$

The updated coefficient vectors for $\boldsymbol{Y}$ are then calculated as,

$$\left[\boldsymbol{c}_1,\ ..,\ \boldsymbol{c}_{i-1},\ \boldsymbol{c}_{i+1},\ \ldots,\ \boldsymbol{c}_N,\ \boldsymbol{c}_i\right] = \boldsymbol{\Sigma}_k \boldsymbol{V}_k \tag{16}$$

where the location of coefficient vectors is reordered. This phenomenon is attributed to the fact that the new snapshot is always attached at the end of the entire snapshot matrix, as shown in Eq. (7).

Update of coefficient model

During the update of POD components in Sects. "Conventional incremental POD" and "Incremental POD with forgetting factor", both the values and dimension of the coefficient vector $\boldsymbol{c}$ for the same input $\boldsymbol{x}$ are changed. In other words, the pre-constructed coefficient model, which defines each coefficient vector $\boldsymbol{c}$ as a function of input $\boldsymbol{x}$, could not be applied anymore after performing IPOD or IPOD-FF. Therefore, the coefficient model needs to be updated simultaneously to 1) maintain the prediction capability on all processed inputs (whose online snapshots are accessible) and 2) improve the prediction performance on other inputs (i.e., unseen points).

Generally, the coefficient model is a multi–input–multi–output regression model, whose update task has been discussed in online learning (Hoi et al., 2021) and incremental learning (Yang et al., 2019). For instance, given some new data, the model parameters in the extreme learning machine could be updated incrementally without training the model from scratch based on all data (Huang et al., 2005; Matias et al., 2015). However, the incremental updating method is designed based on fixed old data, which conflicts with the scenario in the online update of POD-ROM. If state-of-the-art neural networks are applied, online learning could be performed with new data, such as fine-tuning model parameters. Although this strategy could tackle the data insufficiency problem of online data, the fine-tuning process needs some prior information (e.g., training epochs, which part to tune) to guarantee learning performance. Meanwhile, fine-tuning a complex neural network could be expensive, which is often unacceptable for online applications. Moreover, the catastrophic forgetting problem exists in online learning, where the prediction capability gained from the already processed online data diminishes gradually in online updating, which is inconsistent with the task in this paper.

Considering that the online data is at the early stage of online update, constructing a coefficient model only with online data could not provide acceptable prediction capabilities. To tackle the data insufficiency, the auxiliary variable (Kennedy & O'Hagan, 2000; Mifsud et al., 2016) is adopted to utilize both offline and online data when updating the coefficient model in this paper. Specifically, the auxiliary variable "0" is added to the input $\boldsymbol{x}$ (i.e., $[\boldsymbol{x},\ 0]$) to indicate a piece of offline data; the extended input $[\boldsymbol{x},\ 1]$ reflects a piece of online data. Based on the extended inputs and corresponding coefficient vectors, a coefficient model is learned to provide online predictions for unseen points. As finding the optimal modeling technique for the coefficient model is not the research objective in this paper, various modeling techniques are optional, such as RBF, KRG, and ELM, whose integration capability in the proposed online update and application method for DT will be discussed later.

Online performance metrics

When a piece of online snapshot $\boldsymbol{y}_i' \in R^{n_{out}\times 1}$ at the input $\boldsymbol{x}_i$ is measured, the performance of the current DT model on $\boldsymbol{y}_i'$ is reflected by the projection residual and the prediction residual. The projection residual $\Delta\boldsymbol{y}_{i,pro}'$ refers to the difference between the online snapshot $\boldsymbol{y}_i'$ and its projection on the space spanned by current POD bases $\boldsymbol{U}_k$.

$$\Delta\boldsymbol{y}_{i,pro}' = \boldsymbol{y}_i' - \boldsymbol{U}_k\boldsymbol{U}_{\boldsymbol{k}}^T\boldsymbol{y}_i' \tag{17}$$

The prediction residual $\Delta\boldsymbol{y}_{i,pre}'$ is defined as the difference between the online snapshot $\boldsymbol{y}_i'$ and the corresponding prediction $\widehat{\boldsymbol{y}}_i$ obtained from the current DT model, i.e.,

$$\Delta\boldsymbol{y}_{i,pre}' = \boldsymbol{y}_i' - \widehat{\boldsymbol{y}}_i = \boldsymbol{y}_i' - \boldsymbol{U}_k f(\boldsymbol{x}_i) \tag{18}$$

where f is the coefficient model discussed in Sect. "Reduced order model with proper orthogonal decomposition"**.**

From the above formulas, the projection residual is only related to POD bases $\boldsymbol{U}_k$, and the prediction residual is related to the coefficient model f when POD bases are determined. Mathematically, the projection $\boldsymbol{U}_k\boldsymbol{U}_{\boldsymbol{k}}^T\boldsymbol{y}_i'$ and the prediction $\boldsymbol{U}_k f(\boldsymbol{x}_i)$ are located on the same space spanned by POD bases $\boldsymbol{U}_k$, where their residuals have different magnitudes. The comparison between $\Delta\boldsymbol{y}_{i,pro}'$ and $\Delta\boldsymbol{y}_{i,pre}'$ is depicted in Fig. 2, where $\Delta\boldsymbol{y}_{i,pre}' \geq \Delta\boldsymbol{y}_{i,pro}'$ always holds. In other words, the projection residual is the theoretical minimal prediction residual for any online snapshot. As the online update task defined in the paper aims to improve the prediction performance of the offline DT model on the unseen points, the projection residual is first reduced to capture characteristics of the system by the updated POD bases, and then the coefficient models are updated to decrease the prediction residual.

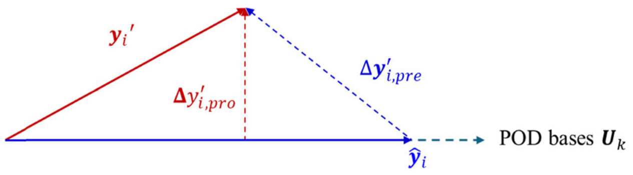


Fig. 2 Comparison of projection residual and prediction residual

Online update and application method

In this paper, a systematic online update and application method is proposed for the POD-ROM model based on IPOD, IPOD-FF, and the coefficient model update. The general diagram is shown in Fig. 3 and the detailed steps are presented below.

Step 1 (Online DT initialization)

Based on the task definition in Sect. "Motivation", an offline DT model has been constructed with the offline dataset ($\boldsymbol{X}$, $\boldsymbol{Y}$) containing samples at N points in the system, where $\boldsymbol{X} = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_N]$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$. The POD computation has been performed to obtain the low-rank POD bases $\boldsymbol{U}_k$, the singular values $\boldsymbol{\Sigma}_k$, and the right singular vectors $\boldsymbol{V}_k$. An offline coefficient model f_{off} has been built to correlate $\boldsymbol{x}_i$ with $\boldsymbol{c}_i$ ($i \in [1, N]$), where $\boldsymbol{c}_i$ is the coefficient vector satisfying $[\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N] = \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T$. Before performing the online update task, all components in the online DT model DT_{on} is initialized as those in the offline one, i.e., $E_{total} = \sum_{i=1}^{N} \|\boldsymbol{c}_i\|^2, \boldsymbol{U}'_k = \boldsymbol{U}_k, \boldsymbol{\Sigma}'_k = \beta_0 \boldsymbol{\Sigma}_k, \boldsymbol{V}'_k = \boldsymbol{V}_k/\beta_0$, $\boldsymbol{c}'_i = \boldsymbol{c}_i$, and $f_{on} = f_{off}$. β_0 is an initial forgetting factor to reduce the overall contribution of offline data. $\boldsymbol{X}' = []$ is the matrix to store the inputs whose online snapshots are used for the online update. $N_{on} = N$ is the number of samples to train the online coefficient model f_{on}.

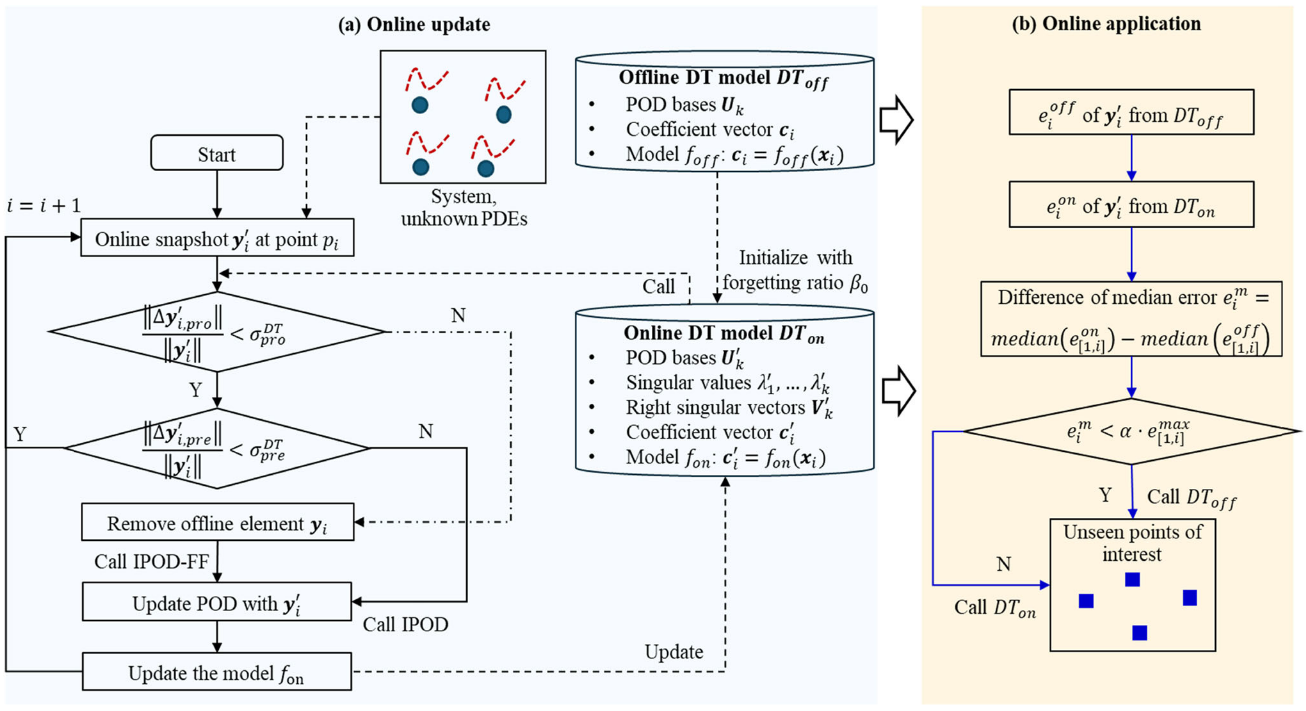
Step 2 (Online performance calculation)

When the online snapshot $\boldsymbol{y}'_i$ at the i-th point with the input $\boldsymbol{x}'_i$ ($\boldsymbol{x}'_i = \boldsymbol{x}_i$ in the task) is measured, the corresponding projection and prediction residuals are calculated as $\Delta \boldsymbol{y}'_{i,\,pro} = \boldsymbol{y}'_i - \boldsymbol{U}'_k (\boldsymbol{U}'_k)^T \boldsymbol{y}'_i$ and $\Delta \boldsymbol{y}'_{i,\,pre} = \boldsymbol{y}'_i - \boldsymbol{U}'_k f_{on}(\boldsymbol{x}'_i)$ based on the current online DT model.

Step 3 (Online update)

Considering that the system PDEs are unknown and so are the responses at the unseen points, the response at the unseen points is only reflected by the DT model prediction. After completing the following sub-steps, the online update process returns to Step 2. The index of point is updated as $i = i + 1$.

Step 3-1 If the normalized projection residual $\|\Delta \boldsymbol{y}'_{i,\,pro}\| / \|\boldsymbol{y}'_i\|$ and the normalized prediction residual $\|\Delta \boldsymbol{y}'_{i,\,pre}\| / \|\boldsymbol{y}'_i\|$ are smaller than the corresponding



(a) Online update
Start
$i = i + 1$
System, unknown PDEs
Offline DT model DT_{off}
POD bases U_k
Coefficient vector c_i
Model f_{off}: $c_i = f_{off}(x_i)$
Online snapshot y'_i at point p_i
Initialize with forgetting ratio β_0
Call
$\frac{\|\Delta y'_{i,pro}\|}{\|y'_i\|} < \sigma^{DT}_{pro}$
N
Y
$\frac{\|\Delta y'_{i,pre}\|}{\|y'_i\|} < \sigma^{DT}_{pre}$
N
Y
Online DT model DT_{on}
POD bases U'_k
Singular values $\lambda'_1, \ldots, \lambda'_k$
Right singular vectors V'_k
Coefficient vector c'_i
Model f_{on}: $c'_i = f_{on}(x_i)$
Remove offline element y_i
Call IPOD-FF
Update POD with y'_i
Call IPOD
Update the model f_{on}
Update
(b) Online application
e_i^{off} of y'_i from DT_{off}
e_i^{on} of y'_i from DT_{on}
Difference of median error $e_i^m = median(e^{on}_{[1,i]}) - median(e^{off}_{[1,i]})$
$e_i^m < \alpha \cdot e^{max}_{[1,i]}$
Y
Call DT_{off}
N
Call DT_{on}
Unseen points of interest

Fig. 3 Flowchart of the proposed online update (**a**) and application method (**b**)

thresholds σ_{pro}^{DT} and σ_{pre}^{DT} respectively, i.e.,

$$\frac{\|\Delta \boldsymbol{y}'_{i,pro}\|}{\|\boldsymbol{y}'_i\|} < \sigma_{pro}^{DT} and \frac{\|\Delta \boldsymbol{y}'_{i,pre}\|}{\|\boldsymbol{y}'_i\|} < \sigma_{pre}^{DT} \tag{19}$$

the current POD bases $\boldsymbol{U}'_k$ and the coefficient model f_{on} are considered to successfully capture the core features of the system and provide an accurate online prediction. Therefore, no update is required.

Step 3-2 If the normalized projection residual $\|\Delta \boldsymbol{y}'_{i,pro}\|/\|\boldsymbol{y}'_i\|$ is larger than σ_{pro}^{DT}, the POD bases $\boldsymbol{U}'_k$ are insufficient to capture the inherent features of the system. The IPOD-FF method is then applied to update POD components to extract features from the online snapshot $\boldsymbol{y}'_i$ and remove elements of $\boldsymbol{y}_i$ from $\boldsymbol{V}'_k$, i.e., $(\boldsymbol{U}'_k, \boldsymbol{\Sigma}'_k, \boldsymbol{V}'_k, E_{total}) = \text{IPOD-FF}(\boldsymbol{U}'_k, \boldsymbol{\Sigma}'_k, \boldsymbol{V}'_k, k, \boldsymbol{y}'_i, \sigma_{pro}^{IPOD}, \sigma_{sv}^{IPOD}, E_{total})$. According to Line 9 in Table 1, the number of POD bases increases by one after performing IPOD-FF, i.e., $k = k + 1$. Meanwhile, the input $\boldsymbol{x}_i$ is moved from $\boldsymbol{X}$ to $\boldsymbol{X}'$, i.e., $\boldsymbol{X}' = \boldsymbol{X} - \boldsymbol{x}_i$, and $\boldsymbol{X}' = [\boldsymbol{X}'; \boldsymbol{x}_i]$. Therefore, the number of training data for f_{on} is unchanged. Based on the updated $\boldsymbol{\Sigma}'_k$ and $\boldsymbol{V}'_k$, the coefficient vectors are then updated as $[\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_{i-1}, \boldsymbol{c}'_{i+1}, \ldots, \boldsymbol{c}'_N, \boldsymbol{c}'_i] = \boldsymbol{\Sigma}'_k(\boldsymbol{V}'_k)^T$. Recalling the discussions on Eq. (9), the coefficient of the online snapshot $\boldsymbol{y}'_i$ on the newly added POD basis is not zero, while the coefficients of all processed offline and online snapshots are close to zero. In other words, the last element in $\boldsymbol{c}'_i$ is a non-zero value, while the last elements in $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_{i-1}, \boldsymbol{c}'_{i+1}, \ldots, \boldsymbol{c}'_N$ are almost zero, as shown in Fig. 4. Such a coefficient structure prohibits constructing an acceptable model to predict reasonable coefficients on the new POD basis for unseen points. To alleviate this issue, the coefficients on the newly added POD basis are excluded when constructing the coefficient model f_{on} based on the method in Sect. "Update of coefficient model". The general modeling purpose is finally formulated as,

$$f_{on}\left(\begin{bmatrix} \boldsymbol{X} & 0 \\ \boldsymbol{X}' & 1 \end{bmatrix}\right) = \left(\boldsymbol{\Sigma}'_k(\boldsymbol{V}'_k)^T\right)_{[1:k-1,:]} \tag{20}$$

where $(\cdot)_{[1:k-1,:]}$ denotes that the former $k-1$ rows in the matrix.

Step 3-3 If $\|\Delta \boldsymbol{y}'_{i,pro}\|/\|\boldsymbol{y}'_i\| < \sigma_{pro}^{DT}$ but $\|\Delta \boldsymbol{y}'_{i,pre}\|/\|\boldsymbol{y}'_i\| > \sigma_{pre}^{DT}$, this step aims to reduce the prediction residual by updating the coefficient models with more online data while maintaining the number of POD bases. The conventional IPOD is adopted here to update POD components with $\boldsymbol{y}'_i$ to find all updated coefficient vectors and keep the orthogonality of $\boldsymbol{V}'_k$ simultaneously, i.e., $(\boldsymbol{U}'_k, \boldsymbol{\Sigma}'_k, \boldsymbol{V}'_k) = \text{IPOD}(\boldsymbol{U}'_k, \boldsymbol{\Sigma}'_k, \boldsymbol{V}'_k, k, \boldsymbol{y}'_i, \sigma_{pro}^{IPOD}, \sigma_{sv}^{IPOD})$. The total energy captured by online POD bases is updated as $E_{total} = E_{total} + \|\boldsymbol{y}'_i\|^2$. The corresponding input $\boldsymbol{x}_i$ is then stored in $\boldsymbol{X}'$ (i.e., $\boldsymbol{X}' = [\boldsymbol{X}'; \boldsymbol{x}_i]$), which means the number of training data for the online coefficient model is increased by one, i.e., $N_{on} = N_{on} + 1$. As the POD bases $\boldsymbol{U}'_k$ are rotated after performing IPOD as shown in Line 12 in Table 1, the coefficients on the last POD basis are not close to zero for all processed snapshots, which means the issue in Fig. 4 does not exist. Therefore, the online coefficient model is learned to predict the coefficients on all POD bases $\boldsymbol{U}'_k$, as shown below.

$$f_{on}\left(\begin{bmatrix} \boldsymbol{X} & 0 \\ \boldsymbol{X}' & 1 \end{bmatrix}\right) = \boldsymbol{\Sigma}'_k(\boldsymbol{V}'_k)^T \tag{21}$$

Step 4 (Online application)

During the online update process, the online DT model could be called at any time to provide predictions for unseen points. Although the online DT model will improve gradually, the online DT model could be worse than the offline DT model at the early stage of the online update. A similar phenomenon has been observed in other applications (Ou et al., 2017; Phalippou et al., 2020). To provide a prediction for the unseen points as acceptable as possible, the performances of both online and offline DT models on the online snapshots are considered to design a criterion. When the online snapshot $\boldsymbol{y}'_i$ is measured, the normalized prediction residuals obtained from online and offline DT models are calculated before performing updating.

$$e_i^{off} = \frac{\|\boldsymbol{y}'_i - DT_{off}(\boldsymbol{x}'_i)\|}{\|\boldsymbol{y}'_i\|},\ e_i^{on} = \frac{\|\boldsymbol{y}'_i - DT_{on}(\boldsymbol{x}'_i)\|}{\|\boldsymbol{y}'_i\|} \tag{22}$$

The median residual is then obtained from stored residual values of all former i online snapshots, denoted as $median(e_{[1,i]}^{off})$ and $median(e_{[1,i]}^{on})$ for offline and online DT models respectively. Their curves are compared in Fig. 5 based on a numerical problem in Sect. "Numerical problems", where the performance of the online DT model on online snapshots increases with the update process. Generally, a smaller median residual indicates that the corresponding DT model would predict better on the unseen points. The difference between two median residuals is calculated as $e_i^m = median\left(e_{[1,i]}^{on}\right) - median(e_{[1,i]}^{off})$. The maximum difference among all former i online snapshots is then defined as $e_{[1,i]}^{max}$. If the following criterion is satisfied,

$$e_i^m < \alpha \cdot e_{[1,i]}^{max} \tag{23}$$

the online DT model is anticipated to provide better performance at the unseen points and the updated online DT

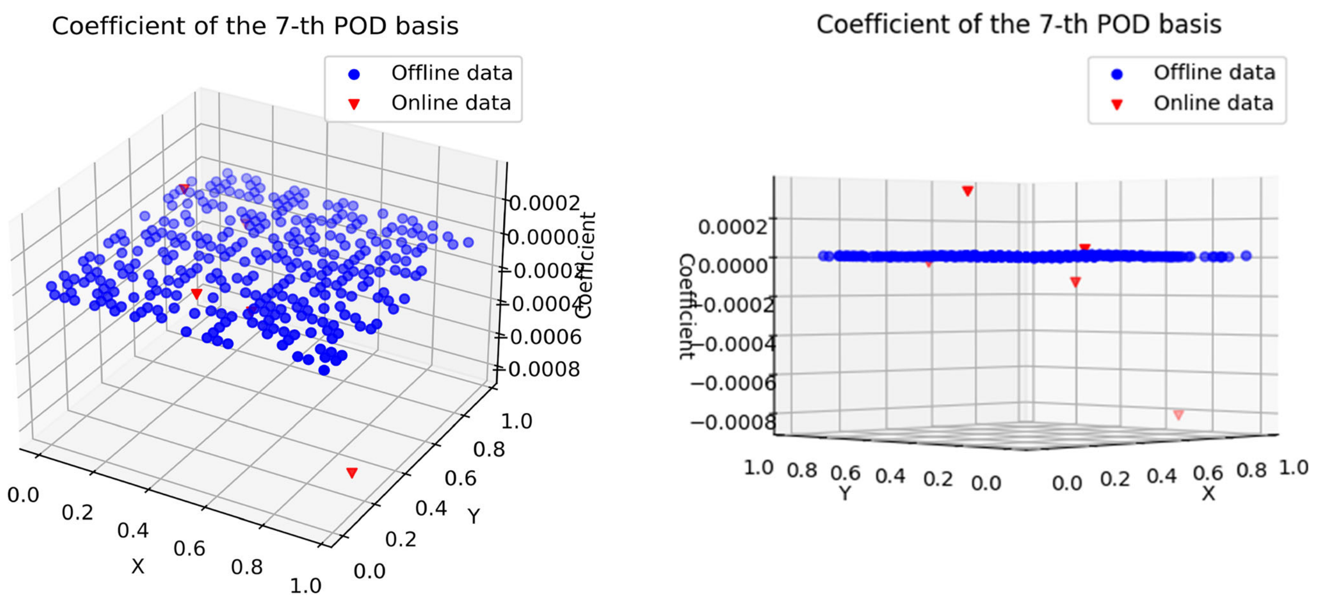


Fig. 4 Coefficients of online and offline snapshots when the number of POD bases is changed. The coefficients of the newly added POD basis are almost zero for all offline data (bule circles) and some processed online data (red triangles). "X" and "Y" refer to the location coordinates of the data in the space (Color figure online)

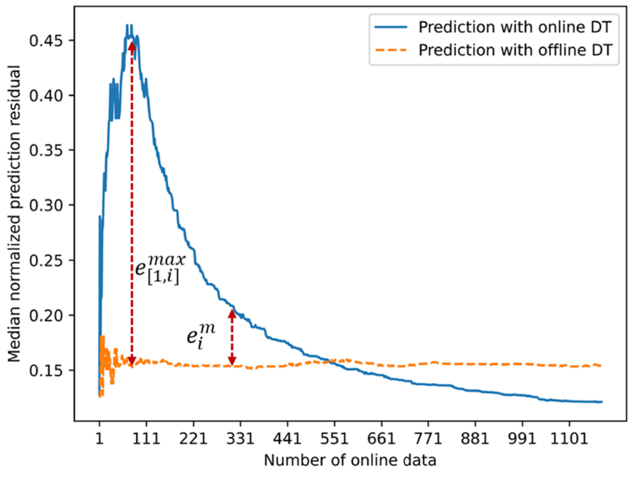


Fig. 5 Median prediction residuals obtained from online and offline DT models

model is called for online application; otherwise, the offline DT model is selected for prediction. The hyperparameter $\alpha \in [0,1)$ controls the proportion of the threshold of e^m_i in the maximum difference. The effects of α will be discussed in Sect. "Discussion and future work".

Numerical problems

In this section, the proposed online update and application method is tested on several numerical problems, which are designed based on nonlinear PDEs. It is to be noted that we assume the PDEs are known to generate the dataset but unknown for DT online update.

Data preparation

Assuming a nonlinear system follows the general formulation,

$$PDEs(g(\boldsymbol{p}, t)|\boldsymbol{G}) = 0 \tag{24}$$

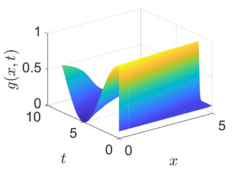
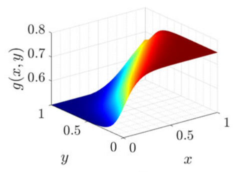
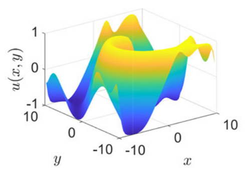
where $g(\boldsymbol{p}, t)$ is the response of the system at the location $\boldsymbol{p} = [p_x, p_y, p_z]$ and the time step $t \in [0, T]$. $PDEs(\cdot)$ contains PDEs, the initial condition equations, and the boundary condition equations, whose parameters are denoted as $\boldsymbol{G}$. Generally, the location satisfies $0 < \boldsymbol{p} < \boldsymbol{L}_p$ where $\boldsymbol{L}_p$ contains the maximum ranges of all dimensions. The total simulation time T is split evenly into N_t time steps, which means the response at the location $\boldsymbol{p}$ during the entire simulation is a vector with a size $N_t \times 1$, i.e., $g(\boldsymbol{p}, \boldsymbol{t}) \in R^{N_t \times 1}$. Meanwhile, the space is discretized as N_p nodes $(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{N_p})$, resulting in N_p response vectors in one completed simulation.

The data preparation process is presented below to generate the required dataset for the test.

Step 1 (Surrogate model for nonlinear PDEs)

Within the range of parameters $\boldsymbol{G}$, N_s different sets of parameters are sampled randomly to perform the simulation. A total

Table 3 General information on nonlinear PDEs and defined numerical problems

Model	Dimension of input $\boldsymbol{x} = [\boldsymbol{p}, \boldsymbol{G}]$	Dimension N_t of the response vector $g(\boldsymbol{p}, \boldsymbol{t})$	Dataset information
 1D Burger's Equation	2	800	Offline dataset: $\boldsymbol{X}_{off} \in R^{200\times 2}$, $\boldsymbol{Y}_{off} \in R^{800\times 200}$ Online dataset: $\boldsymbol{X}_{on} \in R^{200\times 2}$, $\boldsymbol{Y}_{on} \in R^{800\times 200}$ Testing dataset: $\boldsymbol{X}_{test} \in R^{50\times 2}$, $\boldsymbol{Y}_{test} \in R^{800\times 50}$
 2D Burger's Equation at $t = 0.4s$	3	250	Offline dataset: $\boldsymbol{X}_{off} \in R^{313\times 3}$, $\boldsymbol{Y}_{off} \in R^{250\times 313}$ Online dataset: $\boldsymbol{X}_{on} \in R^{313\times 3}$, $\boldsymbol{Y}_{on} \in R^{250\times 313}$ Testing dataset: $\boldsymbol{X}_{test} \in R^{80\times 3}$, $\boldsymbol{Y}_{test} \in R^{250\times 80}$
 2D Reaction-Diffusion Equation at $t = 100s$	5	2000	Offline dataset: $\boldsymbol{X}_{off} \in R^{1000\times 5}$, $\boldsymbol{Y}_{off} \in R^{2000\times 1000}$ Online dataset: $\boldsymbol{X}_{on} \in R^{1000\times 5}$, $\boldsymbol{Y}_{on} \in R^{2000\times 1000}$ Testing dataset: $\boldsymbol{X}_{test} \in R^{250\times 5}$, $\boldsymbol{Y}_{test} \in R^{2000\times 250}$

of $N_s \times N_p$ response vectors are then obtained to train a surrogate model f_{PDEs} to predict the response vector $g(\boldsymbol{p}, \boldsymbol{t})$ according to the location $\boldsymbol{p}$ and the corresponding parameters $\boldsymbol{G}$, i.e., $g(\boldsymbol{p}, \boldsymbol{t}) = f_{PDEs}(\boldsymbol{p}, \boldsymbol{G})$.

Step 2 (Offline, online, and testing dataset generation)

Given a new set of parameters $\boldsymbol{G}_{new}$, the PDEs are solved to obtain the response vector at each point $\boldsymbol{p}_i$, i.e., $g(\boldsymbol{p}_i, \boldsymbol{t})|\boldsymbol{G}_{new}$, $i \in [1, N_p]$. Meanwhile, the predicted response vector is obtained by the surrogate model, i.e., $\widehat{g}(\boldsymbol{p}_i, \boldsymbol{t})|\boldsymbol{G}_{new} = f_{PDEs}(\boldsymbol{p}_i, \boldsymbol{G}_{new})$. In the test, 40% of N_p points (denoted as $\boldsymbol{P}_{40\%}$) are randomly selected to generate the online dataset ($\boldsymbol{X}_{on}$, $\boldsymbol{Y}_{on}$) and the offline dataset ($\boldsymbol{X}_{off}$, $\boldsymbol{Y}_{off}$), and 10% of N_p points (denoted as $\boldsymbol{P}_{10\%}$) are randomly selected as the testing dataset ($\boldsymbol{X}_{test}$, $\boldsymbol{Y}_{test}$). The specific definitions are shown as,

$$\boldsymbol{X}_{off} = \left\{\left[\boldsymbol{p}_j, \boldsymbol{G}_{new}\right]|\boldsymbol{p}_j \in \boldsymbol{P}_{40\%}\right\},$$
$$\boldsymbol{Y}_{off} = \{\widehat{g}\left(\boldsymbol{p}_j, \boldsymbol{t}\right)|\boldsymbol{G}_{new}, \boldsymbol{p}_j \in \boldsymbol{P}_{40\%}\} \tag{25}$$

$$\boldsymbol{X}_{on} = \left\{\left[\boldsymbol{p}_j, \boldsymbol{G}_{new}\right]|\boldsymbol{p}_j \in \boldsymbol{P}_{40\%}\right\},$$
$$\boldsymbol{Y}_{on} = \{g\left(\boldsymbol{p}_j, \boldsymbol{t}\right)|\boldsymbol{G}_{new}, \boldsymbol{p}_j \in \boldsymbol{P}_{40\%}\} \tag{26}$$

$$\boldsymbol{X}_{test} = \left\{\left[\boldsymbol{p}_j, \boldsymbol{G}_{new}\right]|\boldsymbol{p}_j \in \boldsymbol{P}_{10\%}\right\},$$
$$\boldsymbol{Y}_{test} = \{g\left(\boldsymbol{p}_j, \boldsymbol{t}\right)|\boldsymbol{G}_{new}, \boldsymbol{p}_j \in \boldsymbol{P}_{10\%}\} \tag{27}$$

where the input variables $\boldsymbol{x}_j$ at the point $\boldsymbol{p}_j$ are defined as the tuple of the point location and the set of process parameters.

Based on the above process, three nonlinear PDEs with details in the Appendix are selected to define numerical problems, whose general information is summarized in Table 3. More details and the implemented codes of nonlinear PDEs are provided in (Ereiz et al., 2022). In the test, the surrogate model f_{PDEs} for nonlinear PDEs is built by the POD-ROM method where the RBF model is selected as the coefficient model. The SMT library (Saves et al., 2024) is adopted to implement RBF modeling with default settings.

Result comparison

During the online update stage, the normalized projection residual $\|\Delta \boldsymbol{y}_{i,pro}^{on}\|/\|\boldsymbol{y}_i^{on}\|$ and the normalized prediction residual $\|\Delta \boldsymbol{y}_{i,pre}^{on}\|/\|\boldsymbol{y}_i^{on}\|$ of the online snapshot $\boldsymbol{y}_i^{on}$ are obtained from the online DT model. Both metrics are selected to reflect the update process of the online DT model. After updating with the online snapshot $\boldsymbol{y}_i^{on}$, the online application stage is tested on $\boldsymbol{X}^{test}$, whose predicted snapshots are compared with $\boldsymbol{Y}^{test}$. The normalized prediction residual

Table 4 Hyperparameter setting in the test

Hyperparameter	Notation	Value
Energy percentage threshold in POD	ε	$1 - 10^{-6}$
Normalized projection residual threshold in IPOD	σ_{pro}^{IPOD}	0.0001
Singular value threshold in IPOD	σ_{sv}^{IPOD}	10^{-12}(Fareed et al., 2018)
Normalized projection residual threshold in DT updating	σ_{pro}^{DT}	0.0001
Normalized prediction residual threshold in DT updating	σ_{pre}^{DT}	0.001
Initial forgetting factor in DT updating	β_0	0.5
Proportion of the maximum difference in Eq. (23)	α	0.01
Coefficient model option	f	RBF

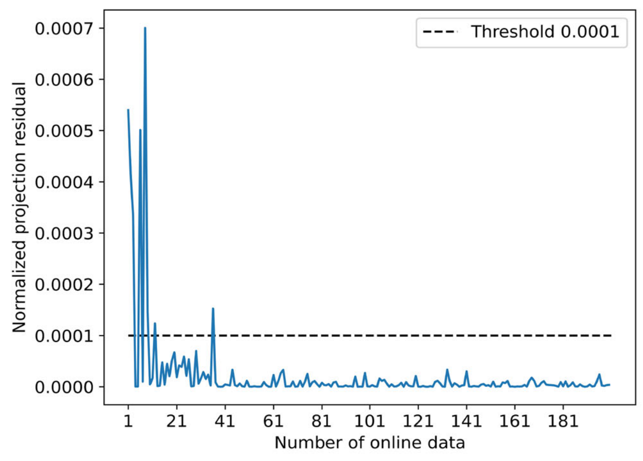
$\|\Delta \boldsymbol{y}_{i,\,pre}^{test}\| / \|\boldsymbol{y}_i^{test}\|$ on each test snapshot is then adopted to demonstrate the performance of the proposed online application strategy. Meanwhile, the median normalized prediction residuals obtained by both offline and online DT models on the online snapshots are compared to study effects of the criterion in Eq. (23), as described in Sect. "Online update and application method". The applied hyperparameter setting in the test is summarized in Table 4, and their effects will be discussed later in Sect. "Discussion and future work".

From results summarized in Fig. 6a, Fig. 7a, and Fig. 8a, it is observed that the normalized projection residuals reduce from a large value to less than the predefined threshold 0.0001 in the three numerical problems. Therefore, the updated POD bases can capture more features gradually with more online data for updates. The unexpected peak values are observed sometimes in the normalized projection residual during the later stage of the online update, which is attributed to the fact that the POD bases updated with current online data cannot guarantee to capture all necessary features for all unseen data. Although the normalized prediction residuals on the online data could converge to a value larger than the threshold 0.001, the common trend in three numerical problems is that the performance of the online DT model is improved gradually during the online update, as shown in Fig. 6b, Fig. 7b, and Fig. 8b. As the normalized projection residual has converged to the acceptable value, the large normalized prediction residual is mainly attributed to the performance of the coefficient model f. In the test, the RBF model, whose performance is affected by the shape coefficient, is selected to construct f as described in Sect. "Update of coefficient model". However, a default value of the shape coefficient is applied in all tests. When comparing both normalized residuals, a better convergence rate is observed in the normalized projection residual in most cases. This demonstrates that during the online update of POD-ROM as the DT model, the accuracy of POD bases is improved first and the performance of the coefficient model is then improved with more online data.

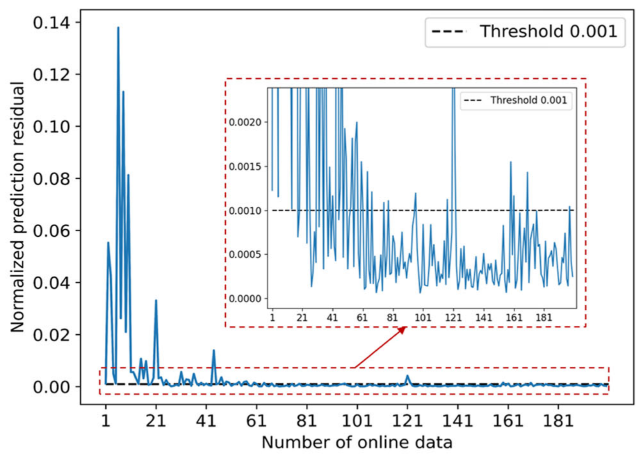
When comparing the median normalized prediction residuals obtained by both offline and online DT models in Fig. 6c, Fig. 7c, and Fig. 8c, it is found that the median value from the offline DT model converges to a plateau after testing on some online data, as the offline DT model is fixed during the online update process. On the contrary, the median value reduces gradually during the entire update process for the online DT model. Another observation is that the median value of the online DT model could be worse than that of the offline DT model, especially at the early stage of the online update.

Based on performance curves of the online data provided by online and offline DT models, the performance on the test data is obtained from online or offline model according to the designed criterion in Eq. (23). Generally, the median normalized prediction residual reduces gradually during the online update process, as shown in Figs. 6d, 7d and 8d. Compared with the baseline performance indicated as the blue dashed line (i.e., no updating and number of online data is 0), the median value reduces from around 0.0020 to around 0.0003 for 1D Burgers' equation, from around 0.0105 to around 0.003 for 2D Burgers' equation, and from around 1.45 to around 0.1 for the 2D reaction–diffusion equation when the online update process terminates. More specifically, for problems whose online DT model is worse than the offline one at the early stage (Figs. 6c and 7c), the offline DT model is applied first, resulting in the constant performance on the test data, as shown in Figs. 6d and 7d. When the online DT model is selected for online application, the performance on the test data improves gradually. For the 2D reaction diffusion equation, the online DT model outperforms the offline one on the online data during the entire process (except for the first online data), as shown in Fig. 8c. The online DT model is applied to the test data, whose performance is summarized in Fig. 8d. Although the median value is larger than the baseline in a few cases, the overall performance of the proposed method is better, which is consistent with the assumption that a better performance on the accessible online data would indicate a better performance on the unseen test data.

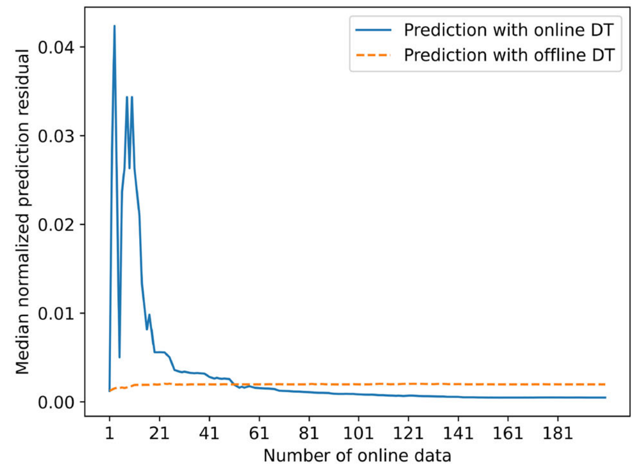
The above discussions demonstrate that by using the proposed online update and application method, the online DT model improves gradually and its performance could converge after a certain number of online data is used for an update in each problem. Meanwhile, the final application performance on unseen data is better than using the offline DT model alone. Therefore, the proposed method is promising to



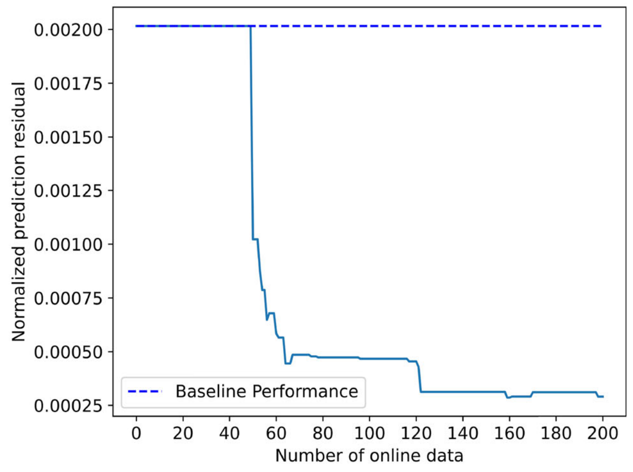
(a) $\|\Delta \boldsymbol{y}_{i,pro}^{on}\|/\|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i^{on}$



(b) $\|\Delta \boldsymbol{y}_{i,pre}^{on}\|/\|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i'$



(c) $median(e_{[1,i]}^{on})$ and $median(e_{[1,i]}^{off})$



(d) Median $\|\Delta \boldsymbol{y}_{i,pre}^{test}\|/\|\boldsymbol{y}_i^{test}\|$ on $\boldsymbol{Y}_{test}$

Fig. 6 Performance of the online update method for 1D Burgers' Equation

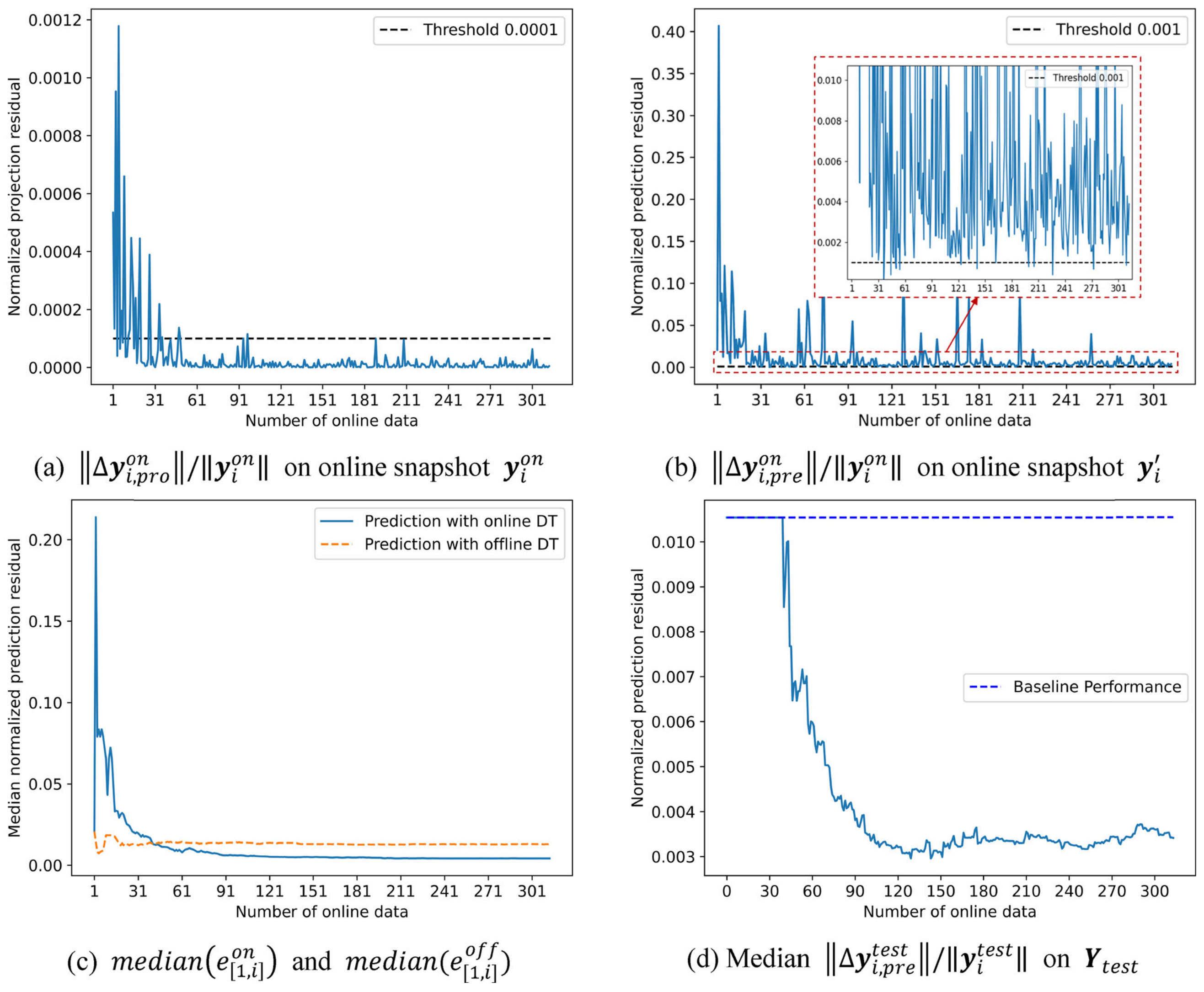
be integrated into online applications (e.g., online control and in-situ monitoring) to provide predictions for unseen points in the system.

Additive manufacturing application

The concept of DT has been applied in additive manufacturing for thermal prediction (Yavari et al., 2021), process monitoring (Shao et al., 2024), and process optimization (Karkaria et al., 2024). More details about DT applications in additive manufacturing are reviewed in (Gaikwad et al., 2020; Mu et al., 2023; Phanden et al., 2022; Su et al., 2023; Zhang et al., 2020). However, current works focus on constructing an accurate offline DT model and applying it to offline tasks (e.g., process optimization) and online tasks (e.g., monitoring) directly. The online update of DT for additive manufacturing is seldom discussed. This section aims to test the applicability of the proposed method in additive manufacturing problems.

Data preparation

Three simulations about directed energy deposition (DED) additive manufacturing are completed with the ANSYS DED module in ANSYS Workbench 2022 R2 to collect the thermal data. The information about the completed simulations is summarized in Table 5, as well as the defined datasets adopted in the online update task. Specifically, a hollow cylinder with the material 17-4PH Stainless Steel is used in the former two simulations, and a cube with the material Inconel 625 is adopted in the third simulation. In each simulation, the



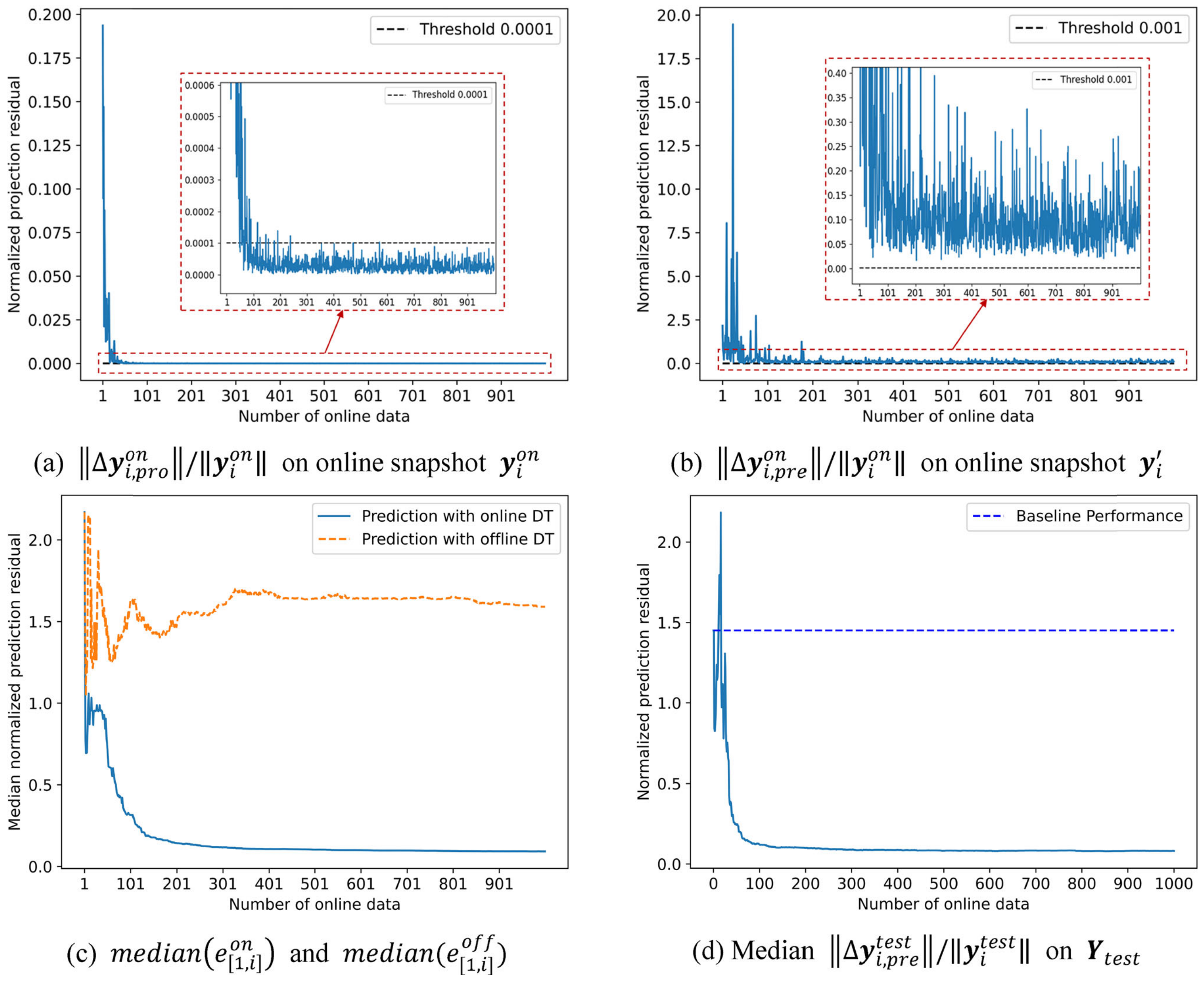
(a) $\|\Delta \boldsymbol{y}_{i,pro}^{on}\| / \|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i^{on}$ (b) $\|\Delta \boldsymbol{y}_{i,pre}^{on}\| / \|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i'$

(c) $median(e_{[1,i]}^{on})$ and $median(e_{[1,i]}^{off})$ (d) Median $\|\Delta \boldsymbol{y}_{i,pre}^{test}\| / \|\boldsymbol{y}_i^{test}\|$ on $\boldsymbol{Y}_{test}$

Fig. 7 Performance of online update method for 2D Burgers' Equation

substrate material is identical to the part material. The properties of 17-4PH Stainless Steel and Inconel 625 could be found in Refs. (Sandmeyer Steel Company, 2016) and (Corrotherm International, 2024), respectively. The thickness of each layer in all geometries is set to 1 mm. The cluster volume is set to 1 mm^3, which determines the volume of the material to be added to the simulation in each time step. The lower the cluster volume, the more computational expensive the simulation. The meshes in simulations are generated by the sweep method with a mesh size of 1 mm and a mesh type of hexahedron. The process temperature is set as 2000 °C in all simulations. ANSYS DED takes the process temperature as an input of the simulation, adds a volume equal to the cluster volume with the temperature equal to the process temperature, and then solves for the rest of the part. The travel speed is set as 10 mm/s in the former two simulations, while it is set as 20 mm/s in the third simulation. The printing pattern in each simulation is shown in Table 5. During the data processing, the temperature data within 10 s after depositing material on each node is selected for study. The time interval is set as 0.01 s. The temperature vector $\boldsymbol{y}_{tem}$ at one node has the size 1000×1.

In each simulation, 60% of all layers are randomly selected to train a surrogate model f_{DED} to predict the temperature vector $\widehat{\boldsymbol{y}}_{tem}^i$ according to i-th node coordinates $\boldsymbol{p}_i = (p_{x_i}, p_{y_i}, p_{z_i})$ and its relative local delay t_i^{delay}, i.e., $\widehat{\boldsymbol{y}}_{tem}^i = f_{DED}(\boldsymbol{p}_i, t_i^{delay})$. The relative local delay of one node on the layer is defined as the time interval between the deposition time of the node and the deposition time of the first deposited node on the same layer. A different layer in the same simulation is then randomly selected to generate datasets required in the defined online update task. On the selected layer, 70% of all nodes ($\boldsymbol{P}_{70\%}^{DED}$) are randomly selected to create the offline



(a) $\|\Delta \boldsymbol{y}_{i,pro}^{on}\| / \|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i^{on}$ (b) $\|\Delta \boldsymbol{y}_{i,pre}^{on}\| / \|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i'$

(c) $median(e_{[1,i]}^{on})$ and $median(e_{[1,i]}^{off})$ (d) Median $\|\Delta \boldsymbol{y}_{i,pre}^{test}\| / \|\boldsymbol{y}_i^{test}\|$ on $\boldsymbol{Y}_{test}$

Fig. 8 Performance of online update method for 2D Reaction Diffusion Equation

and online datasets, while another 10% of all nodes ($\boldsymbol{P}_{10\%}^{DED}$) are selected for the testing dataset. The generated datasets are represented as,

$$\boldsymbol{X}_{off}^{DED} = \left\{\left[\boldsymbol{p}_j, t_i^{delay}\right] | \boldsymbol{p}_j \in \boldsymbol{P}_{70\%}^{DED}\right\},$$
$$\boldsymbol{Y}_{off}^{DED} = \{\widehat{\boldsymbol{y}}_{tem}^i | \boldsymbol{p}_j \in \boldsymbol{P}_{70\%}^{DED}\} \tag{28}$$

$$\boldsymbol{X}_{on}^{DED} = \left\{\left[\boldsymbol{p}_j, t_i^{delay}\right] | \boldsymbol{p}_j \in \boldsymbol{P}_{70\%}^{DED}\right\},$$
$$\boldsymbol{Y}_{on}^{DED} = \{\boldsymbol{y}_{tem}^i | \boldsymbol{p}_j \in \boldsymbol{P}_{70\%}^{DED}\} \tag{29}$$

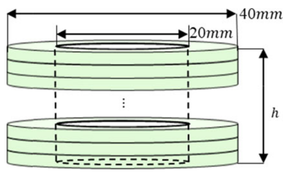
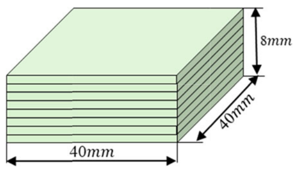
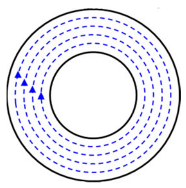
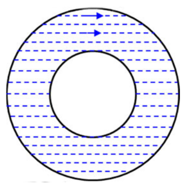
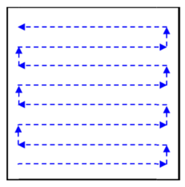
$$\boldsymbol{X}_{test}^{DED} = \left\{\left[\boldsymbol{p}_j, t_i^{delay}\right] | \boldsymbol{p}_j \in \boldsymbol{P}_{10\%}^{DED}\right\},$$
$$\boldsymbol{Y}_{test}^{DED} = \{\boldsymbol{y}_{tem}^i | \boldsymbol{p}_j \in \boldsymbol{P}_{10\%}^{DED}\} \tag{30}$$

where $\boldsymbol{y}_{tem}^i$ is the simulation temperature vector at the node $\boldsymbol{p}_j$. The final generated datasets according to each simulation are summarized at the last row in Table 5.

Result comparison

Using the same setting in Sect. "Result comparison", the proposed method is tested on the three tasks defined on three simulations. Similar to the observations in numerical problems in Sect. "Result comparison", the normalized projection residual converges faster than the normalized prediction residual during the online update process, as shown in Fig. 9a, b, Fig. 10a, b, and Fig. 11a, b. Meanwhile, compared with the baseline performance on the test data, the median normalized prediction residual converges from around 2.9 to around 0.05 in Simulation 1 (Fig. 9d), from around 1.4–0.2 in Simulation 2 (Fig. 10d), and from around 0.14 to around

Table 5 Information about completed DED simulations by ANSYS DED module

Simulation #	1	2	3
Part material	17-4PH Stainless Steel		Inconel 625
Substrate material			
Process temperature	2000°C	2000°C	2000°C
Travel speed	10 mm/s	10 mm/s	20 mm/s
Cluster volume	1 mm^3	1 mm^3	1 mm^3
Number of layers	10	9	8
Layer height	1 mm	1 mm	1 mm
Geometry			
Printing pattern (Dashed lines indicate the used printing trajectories.)			
Dataset information	Offline dataset: $X_{off} \in R^{1050\times4}$ $Y_{off} \in R^{1000\times1050}$ Online dataset: $X_{on} \in R^{1050\times4}$ $Y_{on} \in R^{1000\times1050}$ Testing dataset: $X_{test} \in R^{150\times4}$ $Y_{test} \in R^{1000\times150}$	Offline dataset: $X_{off} \in R^{1211\times4}$ $Y_{off} \in R^{1000\times1211}$ Online dataset: $X_{on} \in R^{1211\times4}$ $Y_{on} \in R^{1000\times1211}$ Testing dataset: $X_{test} \in R^{174\times4}$ $Y_{test} \in R^{1000\times174}$	Offline dataset: $X_{off} \in R^{1176\times4}$ $Y_{off} \in R^{1000\times1176}$ Online dataset: $X_{on} \in R^{1176\times4}$ $Y_{on} \in R^{1000\times1176}$ Testing dataset: $X_{test} \in R^{169\times4}$ $Y_{test} \in R^{1000\times169}$

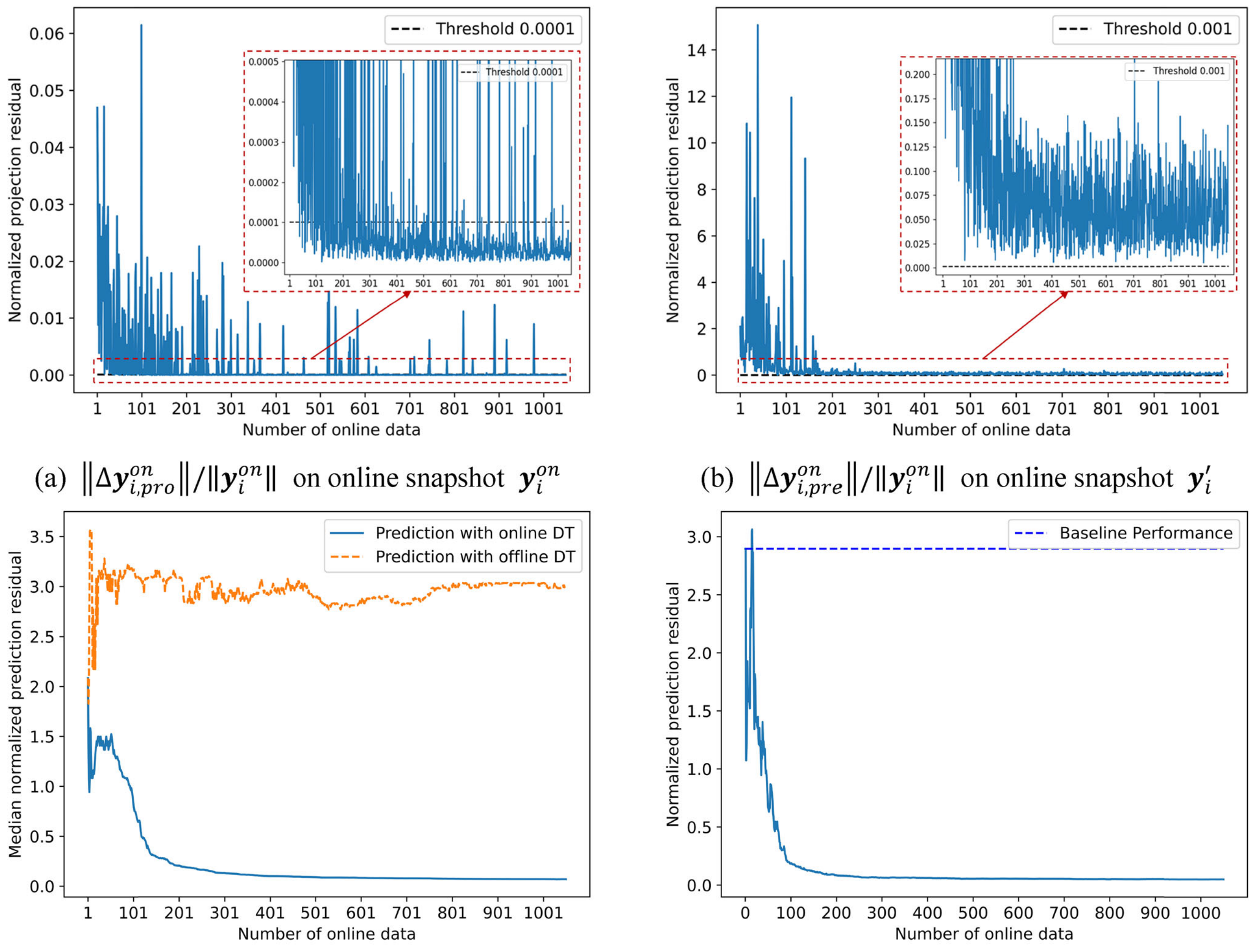
0.105 in Simulation 3 (Fig. 11d). Moreover, when the online DT model consistently outperforms the offline one on the online data (i.e., Simulations 1 and 2), the online DT model is adopted during the online application stage, only with a few worse median values as shown in Figs. 9d and 10d. All tests demonstrate the proposed method is promising in actual engineering applications and outperforms using an offline model without update.

Discussion and future work

Comparison between IPOD and IPOD-FF

In Sect. "Incremental POD with forgetting factor", it is claimed that the proposed IPOD-FF method could update the POD bases more efficiently than the conventional IPOD. Therefore, when the normalized projection is larger than the threshold, the proposed IPOD-FF is applied to update the POD bases instead of the conventional IPOD to update POD bases, as described in Sect. "Online update and application method".

To demonstrate the effectiveness of the proposed IPOD-FF method in updating POD bases, the 1D Burgers' Equation is selected for the study. Based on the collected offline snapshot matrix $\boldsymbol{Y}_{off}$ and the online snapshot matrix $\boldsymbol{Y}_{on}$, the theoretical offline and online POD bases are obtained, denoted as $\boldsymbol{U}^{off}$ and $\boldsymbol{U}^{on}$ respectively. The numbers of bases in $\boldsymbol{U}^{off}$ and $\boldsymbol{U}^{on}$ are counted as n_b^{off} and n_b^{on}, based on which the number of bases for comparison is then calculated as $n_c = \min(n_b^{off}, n_b^{on})$. When performing IPOD and IPOD-FF with the online snapshot sequentially, the updated offline POD bases are denoted as $\boldsymbol{U}^{update}$. During the update process, the former n_c bases $\boldsymbol{u}_{1,\dots,n_c}^{update}$ in $\boldsymbol{U}^{update}$ is compared with the former n_c online POD bases $\boldsymbol{u}_{1,\dots,n_c}^{on}$ in $\boldsymbol{U}^{on}$, where the sum of normalized differences is calculated as $\sum_{i=1}^{n_c} \|\boldsymbol{u}_i^{on} - \boldsymbol{u}_i^{update}\| / \|\boldsymbol{u}_i^{on}\|$.



(a) $\|\Delta \boldsymbol{y}_{i,pro}^{on}\| / \|\boldsymbol{y}_{i}^{on}\|$ on online snapshot $\boldsymbol{y}_{i}^{on}$ (b) $\|\Delta \boldsymbol{y}_{i,pre}^{on}\| / \|\boldsymbol{y}_{i}^{on}\|$ on online snapshot $\boldsymbol{y}_{i}'$

(c) $median(e_{[1,i]}^{on})$ and $median(e_{[1,i]}^{off})$ (d) Median $\|\Delta \boldsymbol{y}_{i,pre}^{test}\| / \|\boldsymbol{y}_{i}^{test}\|$ on $\boldsymbol{Y}_{test}$

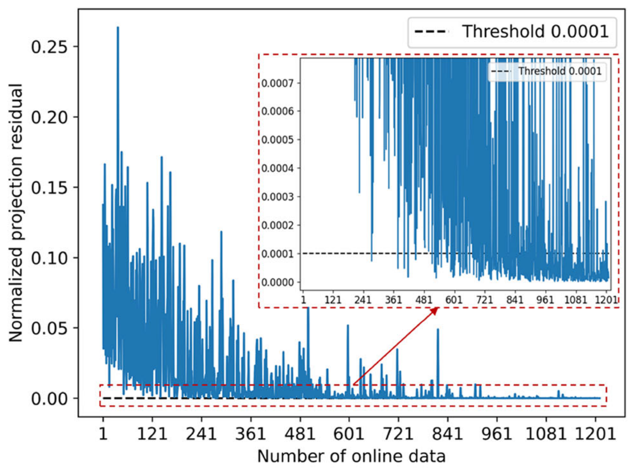
Fig. 9 Performance of online update method in the online task of Simulation 1

The performances obtained by IPOD and IPOD-FF are shown in Fig. 12. Although both methods reduce the normalized difference between the updated offline bases and the ideal online bases gradually, it is observed that IPOD-FF provides a larger reduction rate than IPOD, especially when the number of online data is large. Such observation can also be made when the same initial forgetting factor β_0 is applied to both IPOD-FF and IPOD. Therefore, given a certain number of online data, the proposed IPOD-FF outperforms the conventional IPOD in terms of capturing relevant features.

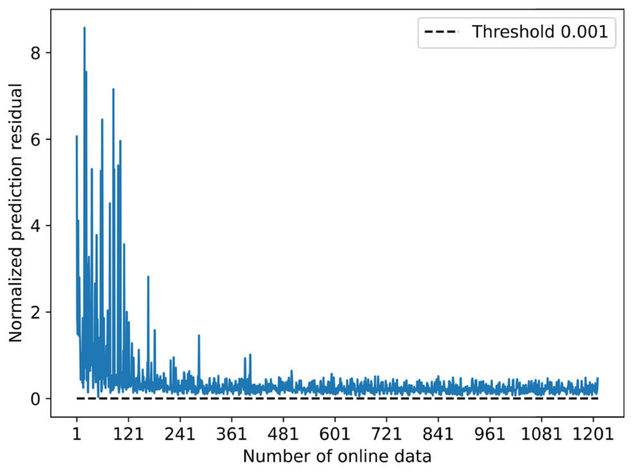
Effects of initial forgetting factors

In the proposed online update and application method, the initial forgetting factor β_0 is adopted to reduce the overall contribution of offline data when performing IPOD or IPOD-FF. Similar to Sect. "Comparison between IPOD and IPOD-FF", the nonlinear PDEs of the 1D Burgers' Equation are adopted to study the effects of the initial forgetting factors.

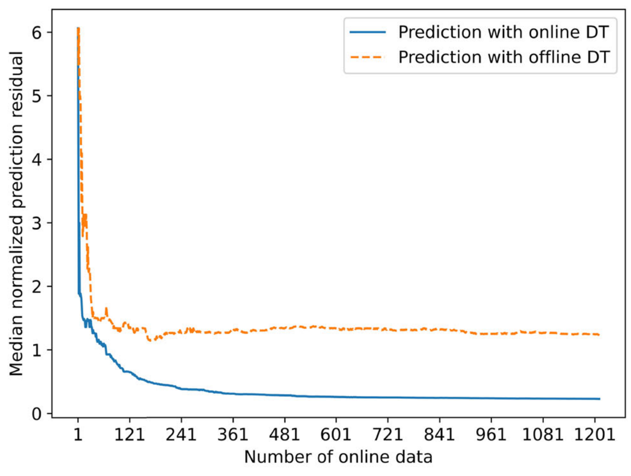
Figure 12 compares the performance (i.e., the sum of normalized differences between the updated offline bases and the ideal online POD bases) obtained by IPOD-FF and IPOD under various initial forgetting factors. It is observed that a smaller initial forgetting factor provides a better performance of IPOD-FF or IPOD. For example, the value reduces from around 0.25 in IPOD-FF without an initial forgetting factor to around 0.05 in IPOD-FF with an initial forgetting factor of 0.3. The value also reduces from around 0.35 to around 0.15 in IPOD when the initial forgetting factor is set as 0.5. Therefore, applying the initial forgetting factor is effective in making the updated POD bases closer to the theoretical ones.



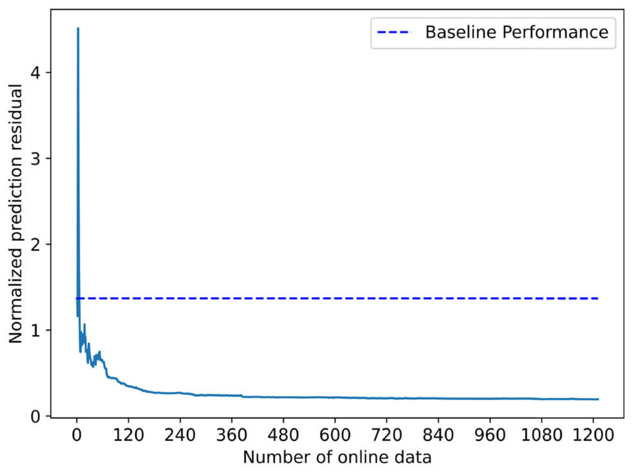
(a) $\|\Delta \boldsymbol{y}_{i,pro}^{on}\| / \|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i^{on}$



(b) $\|\Delta \boldsymbol{y}_{i,pre}^{on}\| / \|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i'$



(c) $median(e_{[1,i]}^{on})$ and $median(e_{[1,i]}^{off})$



(d) Median $\|\Delta \boldsymbol{y}_{i,pre}^{test}\| / \|\boldsymbol{y}_i^{test}\|$ on $\boldsymbol{Y}_{test}$

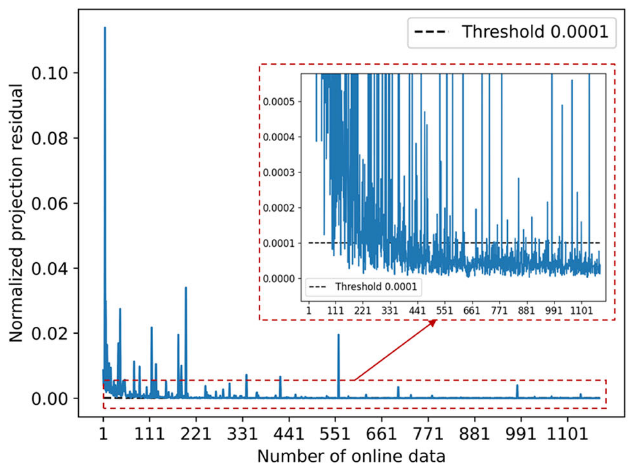
Fig. 10 Performance of the online update method in the online task of Simulation 2

Effects of coefficient model options

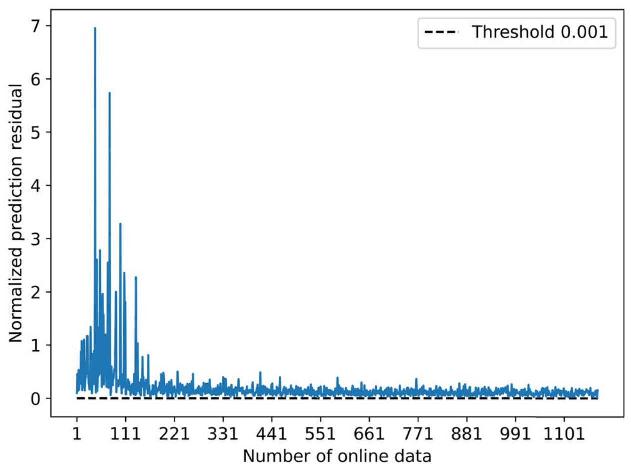
In Sects. "Numerical problems" and "Additive manufacturing application", only RBF is tested as the coefficient model. To demonstrate the generality of the proposed online update and application method, KRG and ELM are selected for study. In the test, KRG and ELM are implemented by the SMT library (Saves et al., 2024) and the *elm* library (Li, 2018) respectively. For ELM, the number of hidden layers is set as $16 \times (n_{in} + k)$, where n_{in} and k are the dimensions of the input x and the corresponding coefficient vector respectively. All hyperparameters of KRG and all other hyperparameters of ELM are set as default values.

Table 6 compares the results of RBF, KRG, and ELM on the 1D Burgers' equation and 2D Burgers' equation. Generally, different coefficient models provide different performances of both online and offline DT models but the differences are relatively insignificant. More specifically, the number of online data applied in Step 3.2 (i.e., large projection residuals) of the proposed method is constant for each problem, while the number of online data adopted in Step 3.3 (i.e., small projection residuals but large prediction residuals) varies with coefficient models. Meanwhile, compared with the median normalized prediction residuals obtained by DT_{off} in both problems with different coefficient models, the corresponding residual values from DT_{on} are always smaller when the online update process is completed.

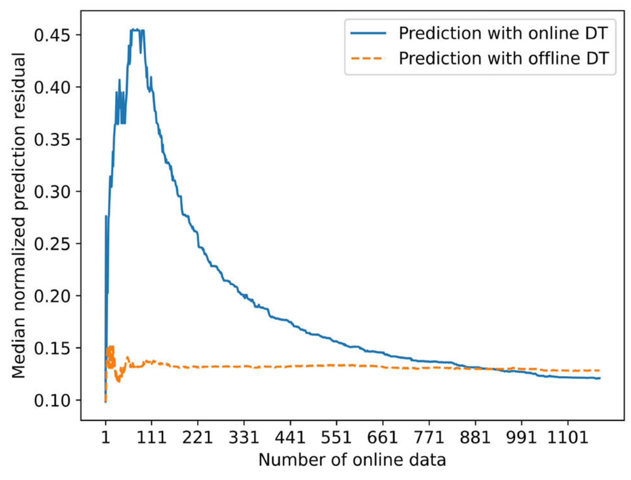
Therefore, the proposed online update and application method could be integrated with various modeling methods for the coefficient model. The coefficient model option only affects the part of the online update stage where more online data is required to get a better coefficient model.



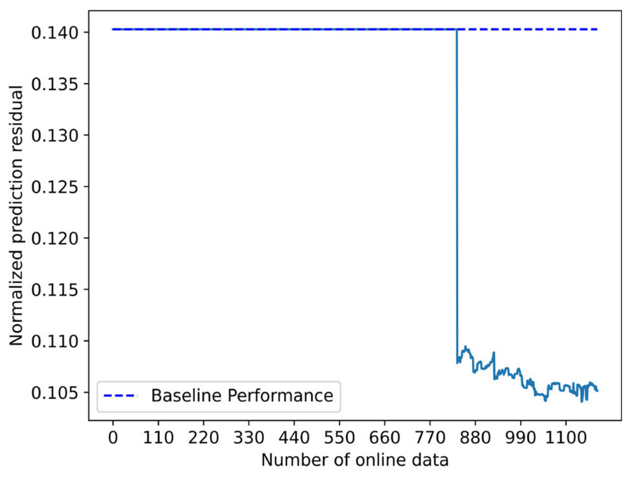
(a) $\|\Delta \boldsymbol{y}_{i,pro}^{on}\| / \|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i^{on}$



(b) $\|\Delta \boldsymbol{y}_{i,pre}^{on}\| / \|\boldsymbol{y}_i^{on}\|$ on online snapshot $\boldsymbol{y}_i'$



(c) $median(e_{[1,i]}^{on})$ and $median(e_{[1,i]}^{off})$



(d) Median $\|\Delta \boldsymbol{y}_{i,pre}^{test}\| / \|\boldsymbol{y}_i^{test}\|$ on $\boldsymbol{Y}_{test}$

Fig. 11 Performance of the online update method in the online task of Simulation 3

Table 6 Comparison with different coefficient model options

Problem	f	$n_{3.2}$	$n_{3.3}$	e_{test}^{off}	e_{test}^{on}
1D Burgers' Equation	RBF	8	44	0.0020	0.0003
	KRG	8	71	0.0020	0.0005
	ELM	8	59	0.0020	0.0001
2D Burgers' Equation	RBF	20	281	0.0105	0.0034
	KRG	20	258	0.0104	0.0062
	ELM	20	272	0.0110	0.0035

$n_{3.2}$ and $n_{3.3}$ are the numbers of online data applied in Step 3.2 and Step 3.2 of the online update stage respectively

e_{test}^{off} and e_{test}^{on} are the median normalized prediction residuals on $\boldsymbol{Y}_{test}$ by the offline DT model and the final updated online DT model respectively

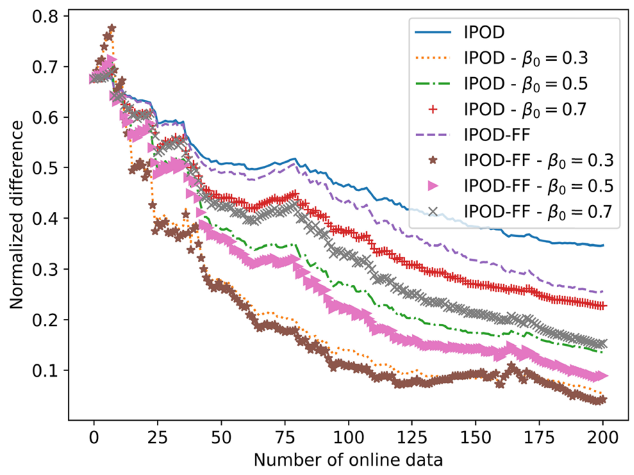


Fig. 12 Comparison between IPOD and IPOD-FF with various forgetting factors

Effects of proportion of max difference

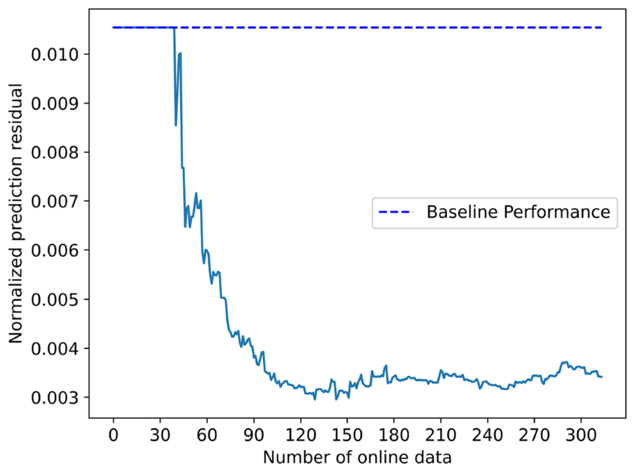
At the online application stage in Sect. "Online update and application method", a hyperparameter α is used to control the proportion of the max median residual difference. Generally, this hyperparameter is independent of the online update stage. It affects only when applying the updated online DT model on the unseen test data. Figure 13 compares the testing performance provided by the online application when using different α values on the 2D Burgers' equation. It is observed that at $\alpha = 0.01$, the median normalized prediction residual of the updated online DT model is on par or lower than the threshold, while for $\alpha = 0.05$, the median values are sometimes larger than the threshold. Therefore, a larger α value would bring a higher risk of worse performance on the test data when applying the online DT model.

Instead of using the proposed proportion of max difference to switch the use of offline and online DT models, another potential option is to adopt a warm-up stage, where the online DT model is always updated. After that, only the online DT is selected during the following online update stage. According to current results, the number of online data, after which the online DT model outperforms the offline one, is around 10 (5%) in 1D Burgers Equation, around 40 (12.8%) in 2D Burgers Equation, around 20 (2%) in 2D Reaction Diffusion, around 20 (1.9%) in Simulation 1, around 10 (0.8%) in Simulation 2, and around 220 (18.7%) in Simulation 3, where the number in the bracket is the proportion of all offline data in each problem. It is observed that the number of required online data varies with problems. Based on prediction accuracies of online data at all update steps, a small number of online data (e.g., 20) could be used in the warm-up stage for problems whose online DT model always outperforms the offline one, such as the 2D Reaction Diffusion Equation case (Fig. 8c), Simulation 1 (Fig. 9c) and Simulation 2 (Fig. 10c). For problems whose offline DT model outperforms the online one at the early stage, such as 1D/2D Burgers Equation (Fig. 6c and Fig. 7c) and Simulation 3 (Fig. 11c), an amount of online data about 20% of the offline data is recommended for the warm-up stage to obtain an online DT model better than the offline one. However, the generality of both rules needs further studies with more engineering problems.

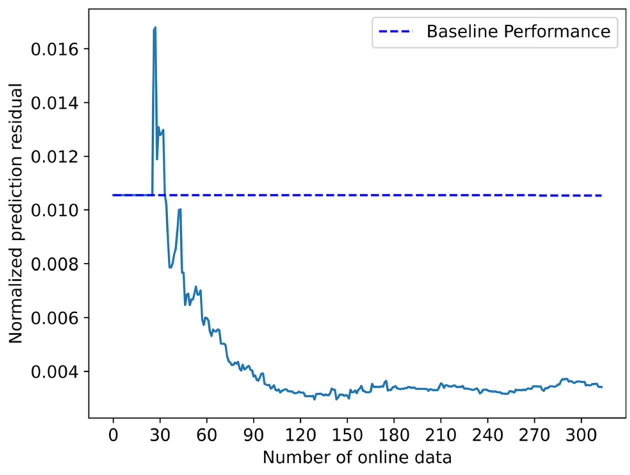
Effects of various thresholds

Several thresholds are applied in the proposed method, including the energy percentage threshold ε in POD, the normalized projection residual threshold σ_{pro}^{IPOD} in IPOD, the singular value threshold σ_{sv}^{IPOD} in IPOD, the normalized projection residual threshold σ_{pro}^{DT} and the normalized prediction residual threshold σ_{pre}^{DT} in DT updating. All of them affect the performance of the proposed online update and application method, in terms of (1) the number of online data selected for online update and (2) the accuracy of the online DT model on the test dataset. Based on 2D Burgers' equation with RBF as the coefficient model, the effects of various thresholds are summarized in Table 7. Several observations are summarized as follows. Details of parameter sensitivity analysis based on the final online prediction residual e_{test}^{on} are shown in Appendix A2.

- The energy percentage threshold ε affects the number of selected POD bases. Generally, a larger ε value will increase the number of selected POD bases, resulting in a better offline model and smaller errors on the test data, as shown in rows 1 and 2 of Table 7.
- The normalized projection residual thresholds σ_{pro}^{IPOD} and σ_{pro}^{DT} control the number of online data to be applied to update POD bases for extracting more features. A larger value relaxes the requirement of POD bases, resulting in fewer required online data. However, a larger value deteriorates the performance of the online DT model on the test data, as shown by the increasing residual values from the first row to rows 3 and 4 in Table 7.
- The singular value threshold σ_{sv}^{IPOD} determines the selection of POD bases. The larger the threshold value, the fewer the POD bases are selected. From the results in rows 1 and 5, it is found that a large σ_{sv}^{IPOD} value would require more online data to update the POD bases, as only several POD bases are kept after online update, which brings a larger projection residual on the next online data. Meanwhile, although more online data is used to update POD bases, the final number of POD bases in the online DT model is still smaller, and the median residual on the test data is larger, indicating a worse performance of the final online DT model.



$\alpha = 0.01$



$\alpha = 0.05$

Fig. 13 Performance on test data when using different α values

Table 7 Effects of various thresholds on 2D Burgers' equation

Id	ε	σ_{pro}^{IPOD}, σ_{pro}^{DT}	σ_{sv}^{IPOD}	σ_{pre}^{DT}	n_{off}^{bases}	$n_{on-ideal}^{bases}$	$n_{3.2}$	$n_{3.3}$	e_{test}^{off}	e_{test}^{on}	$n_{on-final}^{bases}$
1	1-10^{-6}	0.0001	10^{-12}	0.001	4	4	20	281	**0.01054**	**0.00342**	24
2	1-10^{-4}	0.0001	10^{-12}	0.001	2	2	22	279	0.01130	0.00355	24
3	1-10^{-6}	0.001	10^{-12}	0.001	4	4	6	295	0.01054	0.00349	10
4	1-10^{-6}	0.01	10^{-12}	0.001	4	4	0	302	0.01054	0.00379	4
5	1-10^{-6}	0.0001	10^{-2}	0.001	4	4	194	113	0.01054	0.00370	8
6	1-10^{-6}	0.0001	10^{-12}	0.01	4	4	20	66	0.01054	0.00571	24
7	1-10^{-6}	0.0001	10^{-12}	0.1	4	4	20	8	0.01054	0.01054	24
8	1-10^{-6}	0.001	10^{-12}	0.01	4	4	6	74	0.01054	0.00576	10

n_{off}^{bases} is the number of POD bases in the offline DT model

The number $n_{on-ideal}^{bases}$ of ideal online POD bases is obtained from the online snapshot matrix directly

$n_{3.2}$ and $n_{3.3}$ are the numbers of online data applied in Step 3.2 and Step 3.2 of the online update stage respectively

$n_{on-final}^{bases}$ is the number of POD bases in the online DT model when the update is completed on all online data

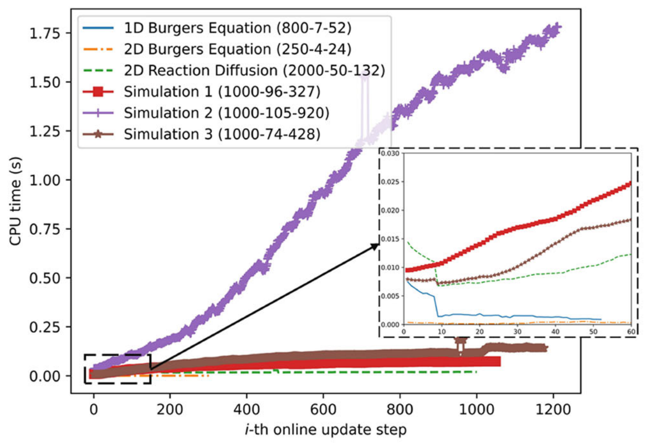
e_{test}^{off} and e_{test}^{on} are the median normalized prediction residuals on $\boldsymbol{Y}_{test}$ by the offline DT model and the final updated online DT model respectively

- The normalized prediction residual threshold σ_{pre}^{DT} only affects the update of the coefficient model, as well as the final performance of the test data. According to the results in rows 1, 6, and 7, a larger σ_{pre}^{DT} value reduces the number of online data used to update the coefficient model. Consequently, the residual of the final online DT model on the test data increases, such as from 0.00342 to 0.01054, which means a worse online DT model is obtained.

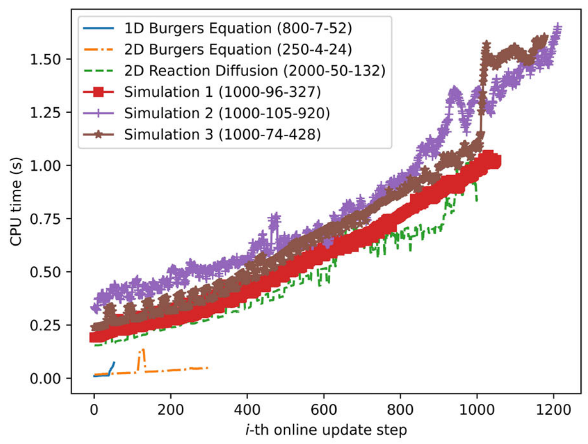
Based on the above discussions, those hyperparameters need careful selection to balance the computation resource required for online updates and the final performance of the online DT model. For instance, when testing the 2D Burgers' equation as shown in row 8 in Table 7, the projection residuals σ_{pro}^{IPOD} and σ_{pro}^{DT}, and the prediction residual σ_{pre}^{DT} could be relaxed simultaneously to get a median residual of 0.00576 with only 80 online data used for update and 10 POD bases in the final online DT model.

Computational efficiency

To study the computational efficiency of the proposed method, the CPU times required for IPOD/IPOD-FF and coefficient model construction with RBF are compared at each online update step. All tests are completed on a low-cost desktop computer (Intel Core i7-3770 CPU @ 3.40 GHz processor, 24.0 GB RAM). After processing the obtained data by the moving average with a window size of 15, the processed time at each online update step is shown in Fig. 14. Generally, both times to perform IPOD/IPOD-FF and construct the



(a) CPU time for performing IPOD/IPOD-FF



(b) CPU time for constructing a coefficient model

Fig. 14 CPU time for performing IPOD/IPOD-FF and constructing a coefficient model with RBF. The numbers (A-B-C) after the benchmark name refer to the problem dimension (**A**), the number of offline POD bases (**B**), and the number of final POD bases (**C**) after completing the online update using the default hyperparameters

coefficient model increase with the number of online update steps in all problems, with following detailed observations.

- The CPU time for POD update is affected by both the problem dimension and the number of selected POD bases. In Fig. 14a, the time for POD update increases from 0.01 s to 0.073 s in the Simulation 1 problem, while the time only increases from 0.015 s to 0.02 s in the 2D Reaction Diffusion problem. Meanwhile, the time increases from 0.036 s to around 1.75 s quickly in the Simulation 2 problem, whose final number of POD bases in the online update is over 900. The reason is that the number of POD bases and problem dimension affect the matrix size to perform decomposition in IPOD methods, as described in Sect. "Conventional incremental POD".
- The time to construct the coefficient model is affected by the number of POD bases and the amount of offline/online data, as they determine the dimension and the number of data points for model construction, as presented in Sect. "Update of coefficient model". For example, the relative ranking of CPU times adopted in different problems is the same as the ranking of number of POD bases in most online update steps, as shown in Fig. 14b.

Based on the above results and discussions, the proposed online update method is computationally efficient in various high-dimensional problems since the total time is less than 2 s in all cases. If the maximum number of required POD bases is smaller than 100, the CPU time is less than 0.002 s for performing IPOD/IPOD-FF and less than 0.1 s for constructing the coefficient model, as shown for both 1D and 2D Burgers Equation problems.

Comparison with transfer learning

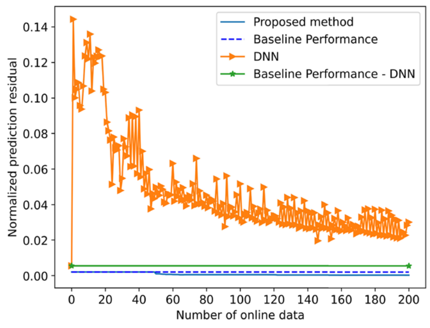
To further explore the performance of the proposed method, the transfer learning methods are adopted for comparison. In transfer learning, a source (or offline) model is firstly constructed. The structure and parameters of the target (online) model are then initialized according to the pre-trained source model, and fine-tuned gradually according to target (online) data sequentially. This idea has been applied in health management (Che et al., 2021), disease recognition (Feng et al., 2023), climate control (Grubinger et al., 2017), additive manufacturing (Sajadi et al., 2024), and other practices, demonstrating its benefits in solving problems with insufficient data and online update tasks for neural networks.

In this section, a deep neural network (DNN) is adopted as the offline DT model, which is trained on the offline dataset $(\boldsymbol{X}_{off}, \boldsymbol{Y}_{off})$. The online DT model is then initialized as the offline one, and fine-tuned based on the online data sequentially. More details about the model structure, the offline training process, and the fine-tuning process are presented in Appendix A3. The results obtained by DNN with transfer learning with the numerical problems are summarized in Table 8. Generally, with more epochs applied when tuning the model online, the CPU required at each step increases, and the final prediction residual e_{test}^{on} on the test data reduces. Compared with the proposed method, tuning a DNN model is more computationally expensive. For example, the average total times (i.e., the sum of time for IPOD and time for constructing a coefficient model) required by the proposed method are less than 0.1 s, 0.1 s, and 1 s in each step for the 1D Burgers Equation, 2D Burgers Equation, and 2D Reaction Diffusion respectively, as shown in Fig. 14. Although the final prediction residual obtained by DNN would be smaller

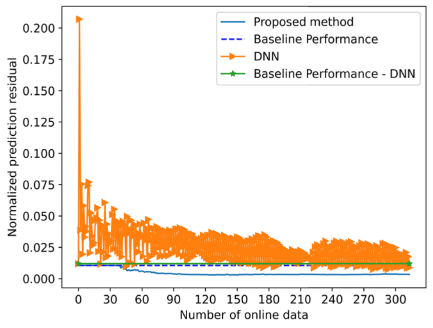
Table 8 Results of DNN on numerical problems

	n_{in}	n_{out}	$epoch_{on}$	$t_{cpu}(s)$	e_{test}^{on}
1D Burgers Equation	2	800	10	0.5007	0.0301
			50	2.4690	0.0033
2D Burgers Equation	3	250	10	0.2196	0.0087
			50	0.8296	0.0021
2D Reaction Diffusion	5	2000	10	4.8339	0.0927
			50	25.3295	0.0264

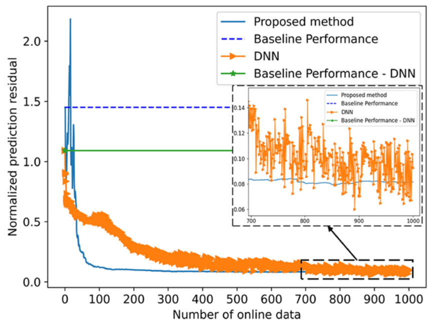
t_{cpu} is the average CPU time applied to tune the DNN model at each online update step
e_{test}^{on} is the median normalized prediction residual on $\boldsymbol{Y}_{test}$ by the final updated DNN model
$epoch_{on}$ is the epoch number applied to tune the model online
n_{in} and n_{out} refers to the input dimension and the output dimension of each problem, respectively



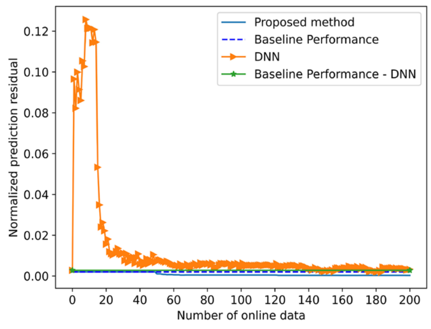
(a) 1D Burgers Equation, $epoch_{on} = 10$



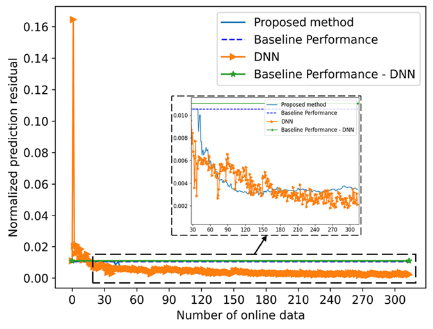
(b) 2D Burgers Equation, $epoch_{on} = 10$



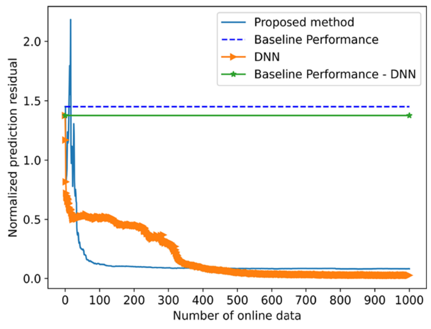
(c) 2D Reaction Diffusion, $epoch_{on} = 10$



(d) 1D Burgers Equation, $epoch_{on} = 50$



(e) 2D Burgers Equation, $epoch_{on} = 50$



(f) 2D Reaction Diffusion, $epoch_{on} = 50$

Fig. 15 Comparison between DNN and the proposed method in numerical methods. The baseline performance refers to the performance of offline DT model on the testing dataset

than the proposed method in a few cases (Fig. 15), the long training time restricts its applications when the time allowed for online update is limited. Meanwhile, the proposed method has a better convergence efficiency, as the performance on the testing data converges with fewer online update steps than DNN. Therefore, the proposed method outperforms DNN using transfer learning in terms of computational and convergence efficiencies in high-dimensional online update tasks.

Although only numerical and additive manufacturing problems are used to test the efficiency and practicability of the proposed method in tackling high-dimensional and PDE-agnostic systems, applications of this method in other engineering scenarios are expected. For instance, when a data-driven ROM model is used for structure monitoring (Ou et al., 2017), the offline ROM model could be updated with the proposed method. Using sensor signals from the structure, the structure health can be monitored in real-time with an updated DT. Meanwhile, in the process industry (Zhu & Ji, 2023), the ROM model with the proposed online update method could be utilized to predict process responses in unseen times, which would be useful for process control. The generality of the proposed method would be tested on those problems in future studies.

Summary

This paper addresses the challenge of updating online digital twins when the partial differential equations of a system are unknown. Without losing generality, the digital twin in this context is built with offline data of some points in the system using a reduced order model and the system is assumed to be a black box. This work proposes an online update method to update the reduced bases and the coefficient models in the offline digital twin model with progressively collected online data of certain points, thereby providing a better performance on unseen points. Two metrics, project and prediction residuals, are defined to indicate when the current reduced bases are insufficient and when the corresponding coefficient models should be updated, respectively. Both offline data and online data are combined for coefficient model construction. One criterion, the difference between median normalized prediction residuals of online and offline digital twins, is defined based on online data to determine whether the online digital twin or the offline one is used on unseen points. To test the performance of the proposed method, three numerical problems are defined based on nonlinear partial differential equations, and three engineering problems are designed based on simulation data using the ANSYS directed energy deposition module. Comparison results demonstrate that the proposed method can gradually improve the prediction performance of the online digital twin model on both online and test data, and provide a final prediction performance on the test data which is no worse than using the offline digital twin model only.

The effects of hyperparameters on the proposed method are examined through a series of simple test studies, Future research will target at eliminating some large residual outliers when applying the online DT model on the test dataset as shown in Sect. “Result comparison” and Sect. ”Result comparison” and testing the proposed methodology in real-world applications.

Appendix

Numerical partial differential equations

(1) 1D Burgers’ equation

The 1D Burgers’ equation is a model for one-dimensional velocity field, whose formula is given as:

$$\frac{\partial g(p_x,t)}{\partial t}+g(x,t)\frac{\partial g(p_x,t)}{\partial p_x}=v\frac{\partial^2 g(p_x,t)}{\partial p_x^2} \tag{31}$$

Initial condition: $g(p_x,0)=a,\ p_x\in[0,L]$
Boundary conditions: $g(0,t)=r(t);\ \frac{\partial g(L,t)}{\partial p_x}=0$

where $g(p_x,t)$ is the velocity of the point p_x at the time$t\in[0,T]$. In the test,$r(t)=0.5\cos\left(\frac{2\pi t}{10}\right)+0.5$, the length$L=5$, the total simulation time$T=8s$, and$v=0.05$. The discrete intervals for simulation time and locations are both set as 0.01, which means there are $N_p=500$ points and $N_t=800$ timesteps in each simulation. Therefore, the response vector at the point p_x in one completed simulation is a vector$g(p_x,\boldsymbol{t})\in R^{800\times 1}$. The parameters $\boldsymbol{G}$ only include the initial velocity in this problem. By changing the initial velocity $a\in[0.01,2]$ with an interval of 0.01, $N_s=100$ simulations are randomly selected to collect data to train a surrogate model for the nonlinear PDEs.

(2) 2D Burgers’ equation

The 2D Burgers’ equation describes a two-dimensional velocity field, whose formula is shown as below, whose boundary conditions can be found in (Ereiz et al., 2022):

$$\begin{aligned}&\frac{\partial g(p_x,p_y,t)}{\partial t}+g(p_x,p_y,t)\frac{\partial g(p_x,p_y,t)}{\partial p_x}\\&\quad+h(p_x,p_y,t)\frac{\partial g(p_x,p_y,t)}{\partial p_y}\\&\quad=v\frac{\partial^2 g(p_x,p_y,t)}{\partial p_x^2}+\frac{\partial^2 g(p_x,p_y,t)}{\partial p_y^2}\\&\quad\frac{\partial h(p_x,p_y,t)}{\partial t}+g(p_x,p_y,t)\frac{\partial h(p_x,p_y,t)}{\partial p_x}\\&\quad+h(p_x,p_y,t)\frac{\partial h(p_x,p_y,t)}{\partial p_y}\\&\quad=v\frac{\partial^2 h(p_x,p_y,t)}{\partial p_x^2}+\frac{\partial^2 g(p_x,p_y,t)}{\partial p_y^2}\end{aligned} \tag{32}$$

Initial conditions: $g(p_x,p_y,0)=\phi_1(p_x,p_y)$, $h(p_x,p_y,0)=\phi_2(p_x,p_y),\ x\in[0,L_x],\ y\in[0,L_y]$

where $g(p_x,p_y,t)$and $h(p_x,p_y,t)$are two velocity components of the point $\boldsymbol{p}=[p_x,p_y]$at time t. In this test, the applied initial conditions are shown as:

$$\begin{aligned}\phi_1(p_x,p_y)&=\frac{3}{4}-\frac{1}{4\left[1+e^{\frac{(-4p_x+4p_y)Re}{32}}\right]},\\\phi_2(p_x,p_y)&=\frac{3}{4}+\frac{1}{4\left[1+e^{\frac{(-4p_x+4p_y)Re}{32}}\right]}\end{aligned} \tag{33}$$

where $Re=\frac{1}{v}$is the Reynold’s number. $L_x=L_y=1$. There are 28 discrete nodes in each direction, resulting in a total of $N_p=784$points in the velocity field. The total simulation time T is 8 s with $N_s=250$discrete timesteps. Therefore, the velocity components of the pint $\boldsymbol{p}=[p_x,$

p_y]in one completed simulation are $g\left(p_x,\ p_y,\ \boldsymbol{t}\right) \in R^{250\times 1}$ and $h\left(p_x,\ p_y,\ \boldsymbol{t}\right) \in R^{250\times 1}$. In this paper, only the velocity component $g\left(p_x,\ p_y,\ \boldsymbol{t}\right)$is studied. The parameters $\boldsymbol{G}$only include the Revalue. By sampling different Revalues within [100, 1000] with an interval of 50, $N_s = 10$simulations are randomly selected to generate data to train a surrogate model for the nonlinear system.

(3) 2D reaction–diffusion equation

The dynamics of the 2D reaction–diffusion PDEs are shown as below:

$$\begin{aligned}\frac{\partial u\left(p_x,\ p_y,\ t\right)}{\partial t} &= \left(1-\left(u^2\left(p_x,\ p_y,\ t\right)+v^2\left(p_x,\ p_y,\ t\right)\right)\right)u\left(p_x,\ p_y,\ t\right)+\beta\left(u^2\left(p_x,\ p_y,\ t\right)+v^2\left(p_x,\ p_y,\ t\right)\right)v\left(p_x,\ p_y,\ t\right)+c_1\left(\frac{\partial^2 u\left(p_x,\ p_y,\ t\right)}{\partial p_x^2}+\frac{\partial^2 u\left(p_x,\ p_y,\ t\right)}{\partial p_y^2}\right)\\ \frac{\partial v\left(p_x,\ p_y,\ t\right)}{\partial t} &= \left(1-\left(u^2\left(p_x,\ p_y,\ t\right)+v^2\left(p_x,\ p_y,\ t\right)\right)\right)v\left(p_x,\ p_y,\ t\right)-\beta\left(u^2\left(p_x,\ p_y,\ t\right)+v^2\left(p_x,\ p_y,\ t\right)\right)u\left(p_x,\ p_y,\ t\right)+c_2\left(\frac{\partial^2 v\left(p_x,\ p_y,\ t\right)}{\partial p_x^2}+\frac{\partial^2 v\left(p_x,\ p_y,\ t\right)}{\partial p_y^2}\right)\end{aligned} \tag{34}$$

where c_1, c_2, and β are three parameters to control the dynamics, which means $\boldsymbol{G} = [c_1,\ c_2,\ \beta]$ in the test. In each simulation, 50 nodes exist in each direction $p_x,\ p_y \in [0,\ 20]$, and $N_t = 2000$ timesteps are generated in the total simulation time $T = 100$ seconds. As a result, the response vectors $u(p_x,\ p_y,\ \boldsymbol{t})$ and $v(p_x,\ p_y,\ \boldsymbol{t})$ at the point $\boldsymbol{p} = [p_x,\ p_y]$ have the size of 2000×1, and a total of $N_p = 2500$ discrete points exist in the field. In this test, only $u(p_x,\ p_y,\ \boldsymbol{t})$ is selected for study. Similar to the previous two problems, $N_s = 200$ simulations are completed with different parameter values within their respective bounds $c_1 \in [0.1,\ 1]$, $c_2 \in [0.1,\ 1]$, and $\beta \in [0.5,\ 2]$ to train a surrogate model for the nonlinear PDEs.

Parameter sensitivity analysis

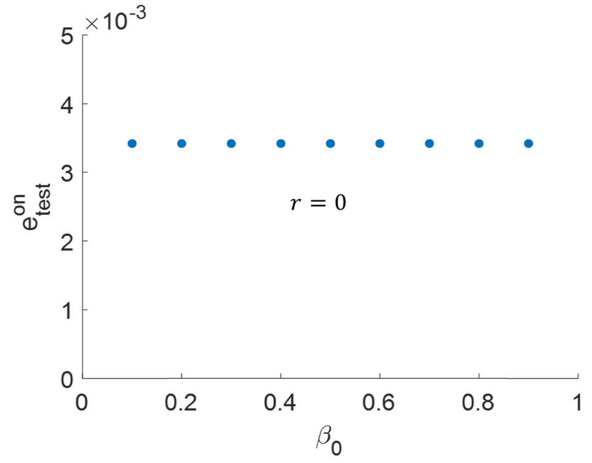
The parameter sensitivity analysis is performed based on the 2D Burgers' Equation problem as defined in Sect. "Data preparation", to study the effects of various hyperparameters on the final DT model performance. In this section, only the median normalized prediction residuals e_{test}^{on} on testing dataset by the final updated online DT model is selected as the performance index. A smaller e_{test}^{on} value indicates to a better DT model. The scatter plot is adopted to demonstrate effects of hyperparameters qualitatively, and the Pearson correlation coefficient r is used to identify the linear correlation between each hyperparameter and the final performance quantitatively (Hamby, 1994).

The local sensitivity analysis is performed for each hyperparameter, where one hyperparameter varies while keeping others constant. The scatter plots and corresponding r values of all hyperparameters are summarized in Fig. 16, with following observations.

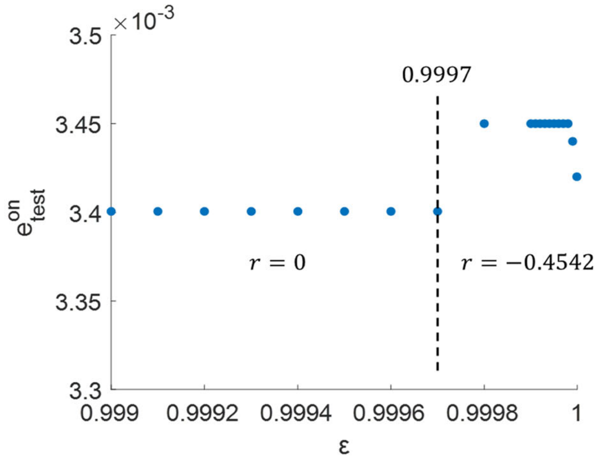
- The initial forgetting factor β_0 is not correlated with e_{test}^{on}. The reason is that the concept "forgetting factor" was proposed to improve the efficiency to capture the latent features from online data, as discussed in Sect. "Effects of initial forgetting factors". Among testes with different β_0 values, most online data is used to update to improve the accuracy of coefficient model, which reduces the effects of different updated POD bases on final model performance.
- For 2D Burgers Equation, each hyperparameter only affects e_{test}^{on} when located in a certain range, such as POD energy threshold $\varepsilon > 0.9997$, projection residual threshold $\sigma_{pro} < 0.003$, singular value threshod $10^{-4} < \sigma_{sv}^{IPO} < 0.03$, and prediction residual threshold $\sigma_{pre}^{DT} < 0.07$. Those ranges would vary with problems. For example, in the 1D Burgers Equation problem, the prediction residual threshold σ_{pre}^{DT} is correlated with e_{test}^{on} only when $\sigma_{pre}^{DT} < 0.007$.
- Generally, the POD energy threshold has a negative correlation with e_{test}^{on}, while a positive correlation is observed in the projection residual threshold, the singular value threshold, and the prediction residual threshold.

Transfer learning

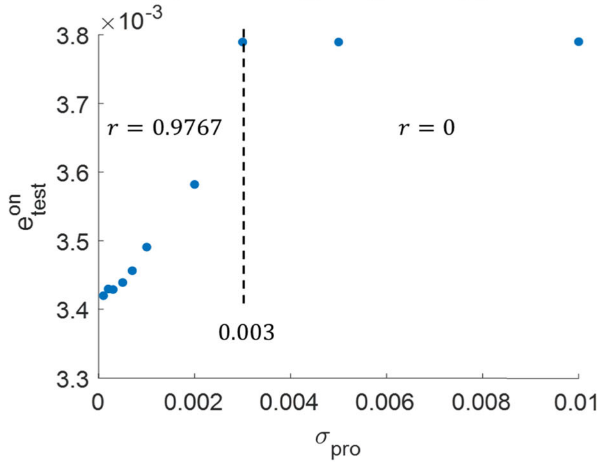
In this study, the fully connected neural network is selected as the deep neural network (DNN), which is used in transfer learning. Considering that the output dimension of defined problems is much higher than the input dimension, the structure shown in Fig. 17 is adopted after several trials to provide acceptable testing performance. Both offline and online training processes follow our previous work (Tang et al., 2024). The pretrained offline DNN model is used to initialize the online DNN model, whose weights and biases are tuned online. The Adam optimizer is used to learn model weights and biases. The learning rate for offline training is set as 0.005 with an epoch number of 1000. The learning rate for online fine tuning is set as 0.001. At the i-th online update step, DNN is tuned based on all accessible online data to



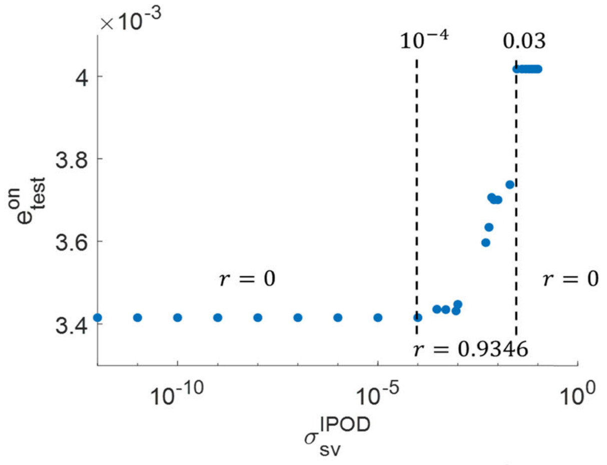
(a) Initial forgetting ratio β_0



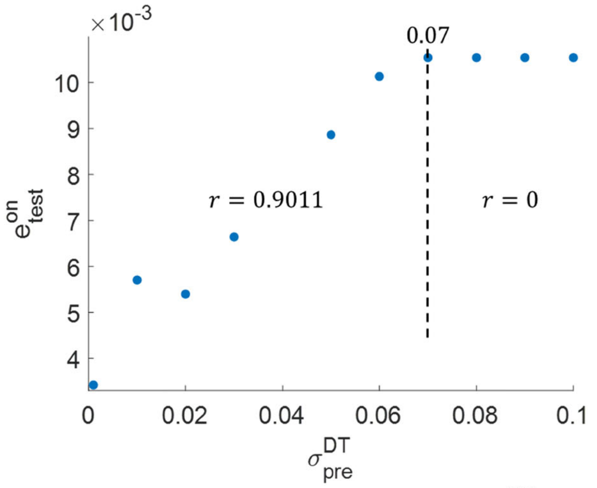
(b) POD energy threshold ε



(c) Projection residual threshold σ_{pro}



(d) Singular value threshold σ_{sv}^{IPOD}



(e) Prediction residual threshold σ_{pre}^{DT}

Fig. 16 Scatter plots and Pearson correlation coefficient r of each hyperparameter

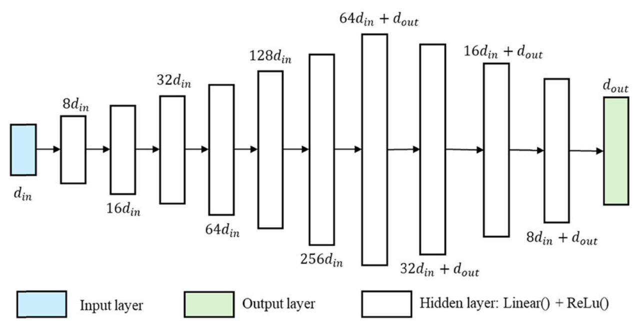


Fig. 17 Model structure of deep neural network in transfer learning. d_{in} is the input dimension. d_{out} is the output dimension. Each hidden layer consists of a Linear() layer and an activation function ReLu(). All codes are implemented by Pytorch

reduce the risk of catastrophic forgetting, where the information learned from previous online data diminish gradually from the model. If DNN is only tuned with current online data, it is observed that the final performance on the testing dataset is worse than the offline DNN model during tests.

Acknowledgements Funding from Natural Science and Engineering Research Council (NSERC) of Canada under the project RGPIN-2019–06601, as well Graduate Dean's Entrance Scholarship (GDES) from Simon Fraser University, is appreciated.

Data availability The dataset is available upon reasonable request from the authors.

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

References

Al-Subaihawi, S., Ricles, J. M., & Quiel, S. E. (2022). Online explicit model updating of nonlinear viscous dampers for real time hybrid simulation. *Soil Dynamics and Earthquake Engineering, 154*, 107108. https://doi.org/10.1016/j.soildyn.2021.107108

Badeau, R., Richard, G., & David, B. (2004). Sliding window adaptive SVD algorithms. *IEEE Transactions on Signal Processing, 52*(1), 1–10. https://doi.org/10.1109/TSP.2003.820069

Brand, M. (2002). Incremental singular value decomposition of uncertain data with missing values. *Lecture Notes in Computer Science, 2350*, 707–720. https://doi.org/10.1007/3-540-47969-4_47

Cardot, H., & Degras, D. (2018). Online principal component analysis in high dimension: Which algorithm to choose? *International Statistical Review, 86*(1), 29–50. https://doi.org/10.1111/insr.12220

Che, Y., Deng, Z., Lin, X., Hu, L., & Hu, X. (2021). Predictive battery health management with transfer learning and online model correction. *IEEE Transactions on Vehicular Technology, 70*(2), 1269–1277. https://doi.org/10.1109/TVT.2021.3055811

Cheng, S., Quilodran-Casas, C., Ouala, S., Farchi, A., Liu, C., Tandeo, P., et al. (2023). Machine learning with data assimilation and uncertainty quantification for dynamical systems: A review. *IEEE/CAA Journal of Automatica Sinica, 10*(6), 1361–1387. https://doi.org/10.1109/JAS.2023.123537

Corrotherm International. (2024). Alloy 625 / Inconel 625. https://www.corrotherm.co.uk/grades/inconel-625. Accessed 11 March 2024

Dorosti, M. (2017). *Reduced-order model updating for prediction of performance variables in mechanical structures*. Eindhoven University of Technology. Retrieved from https://research.tue.nl/files/68595415/20170614_Dorosti.pdf

Ebrahimzadeh Hassanabadi, M., Heidarpour, A., Eftekhar Azam, S., & Arashpour, M. (2020). Recursive principal component analysis for model order reduction with application in nonlinear Bayesian filtering. *Computer Methods in Applied Mechanics and Engineering, 371*, 113334. https://doi.org/10.1016/j.cma.2020.113334

Ebrahimzadeh Hassanabadi, M., Liu, Z., Eftekhar Azam, S., & Dias-da-Costa, D. (2023). A linear Bayesian filter for input and state estimation of structural systems. *Computer-Aided Civil and Infrastructure Engineering, 38*(13), 1749–1766. https://doi.org/10.1111/mice.12973

Eftekhar Azam, S., & Mariani, S. (2018). Online damage detection in structural systems via dynamic inverse analysis: A recursive Bayesian approach. *Engineering Structures, 159*, 28–45. https://doi.org/10.1016/j.engstruct.2017.12.031

Eftekhar Azam, S., Mariani, S., & Attari, N. K. A. (2017). Online damage detection via a synergy of proper orthogonal decomposition and recursive Bayesian filters. *Nonlinear Dynamics, 89*(2), 1489–1511. https://doi.org/10.1007/s11071-017-3530-1

Elshenawy, L. M., Yin, S., Naik, A. S., & Ding, S. X. (2010). Efficient recursive principal component analysis algorithms for process monitoring. *Industrial & Engineering Chemistry Research, 49*(1), 252–259. https://doi.org/10.1021/ie900720w

Altair Engineering. (2023). *2023 Global digital twin survey report vertical breakdown: manufacturing*. https://altair.com/docs/default-source/pdfs/Altair_Global-Survey-Report-Manufacturing-web.pdf

Ereiz, S., Duvnjak, I., & Fernando Jiménez-Alonso, J. (2022). Review of finite element model updating methods for structural applications. *Structures, 41*, 684–723. https://doi.org/10.1016/j.istruc.2022.05.041

Fareed, H., Singler, J. R., Zhang, Y., & Shen, J. (2018). Incremental proper orthogonal decomposition for PDE simulation data. *Computers and Mathematics with Applications, 75*(6), 1942–1960. https://doi.org/10.1016/j.camwa.2017.09.012

Feng, Q., Xu, P., Ma, D., Lan, G., Wang, F., Wang, D., & Yun, Y. (2023). Online recognition of peanut leaf diseases based on the data balance algorithm and deep transfer learning. *Precision Agriculture, 24*(2), 560–586. https://doi.org/10.1007/s11119-022-09959-3

Gaikwad, A., Yavari, R., Montazeri, M., Cole, K., Bian, L., & Rao, P. (2020). Toward the digital twin of additive manufacturing: Integrating thermal simulations, sensing, and analytics to detect process faults. *IISE Transactions, 52*(11), 1204–1217. https://doi.org/10.1080/24725854.2019.1701753

Garbo, A., & Bekemeyer, P. (2022). Unsteady physics-based reduced order modeling for large-scale compressible aerodynamic applications. *Computers and Fluids, 239*, 105385. https://doi.org/10.1016/j.compfluid.2022.105385

Grieves, M. (2014). *Digital twin: manufacturing excellence through virtual factory replication*. https://www.researchgate.net/publication/275211047

Griffiths, L. M., Gaitonde, A. L., Jones, D. P., & Friswell, M. I. (2018). Updating of aerodynamic reduced order models generated using computational fluid dynamics. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 232*(9), 1739–1763. https://doi.org/10.1177/0954410017716698

Grubinger, T., Chasparis, G. C., & Natschläger, T. (2017). Generalized online transfer learning for climate control in residential buildings. *Energy and Buildings, 139*, 63–71. https://doi.org/10.1016/j.enbuild.2016.12.074

Hamby, D. M. (1994). A review of techniques for parameter sensitivity. *Environmental Monitoring and Assessment*, *32*, 135–154. https://deepblue.lib.umich.edu/bitstream/handle/2027.42/42691/10661_2004_Article_BF00547132.pdf?sequence=1

Han, Y., Huang, G., Song, S., Yang, L., Wang, H., & Wang, Y. (2022). Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(11), 7436–7456. https://doi.org/10.1109/TPAMI.2021.3117837

Hoi, S. C. H., Sahoo, D., Lu, J., & Zhao, P. (2021). Online learning: A comprehensive survey. *Neurocomputing, 459*, 249–289. https://doi.org/10.1016/j.neucom.2021.04.112

Huang, G. B., Liang, N. Y., Rong, H. J., Saratchandran, P., & Sundararajan, N. (2005). On-line sequential extreme learning machine. *IASTED International Conference on Computational Intelligence* (pp. 232–237). ACTA Press.

Karkaria, V., Goeckner, A., Zha, R., Chen, J., Zhang, J., Zhu, Q., et al. (2024). Towards a digital twin framework in additive manufacturing: Machine learning and bayesian optimization for time series process optimization. *Journal of Manufacturing Systems, 75*, 322–332. https://doi.org/10.1016/j.jmsy.2024.04.023

Kennedy, M. C., & O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika, 87*(1), 1–13. https://doi.org/10.1093/biomet/87.1.1

Krishnamurthi, R., Kumar, A., Gopinathan, D., Nayyar, A., & Qureshi, B. (2020). An overview of iot sensor data processing, fusion, and analysis techniques. *Sensors (Switzerland), 20*(21), 1–23. https://doi.org/10.3390/s20216076

Li, X. (2018). Extreme learning machine(ELM): Python code. https://github.com/5663015/elm. Accessed 27 April 2023

Li, W., Yue, H. H., Valle-Cervantes, S., & Qin, S. J. (2000). Recursive PCA for adaptive process monitoring. *Journal of Process Control, 10*(5), 471–486. https://doi.org/10.1016/S0959-1524(00)00022-6

Li, X. D., Hulshoff, S., & Hickel, S. (2022). An enhanced algorithm for online proper orthogonal decomposition and its parallelization for unsteady simulations. *Computers and Mathematics with Applications, 126*, 43–59. https://doi.org/10.1016/j.camwa.2022.09.007

Lim, K. Y. H., Zheng, P., & Chen, C. H. (2020). A state-of-the-art survey of digital twin: Techniques, engineering product lifecycle management and business innovation perspectives. *Journal of Intelligent Manufacturing, 31*(6), 1313–1337. https://doi.org/10.1007/s10845-019-01512-w

Liu, M., Fang, S., Dong, H., & Xu, C. (2021). Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems, 58*, 346–361. https://doi.org/10.1016/j.jmsy.2020.06.017

Lu, K., Zhang, K., Zhang, H., Gu, X., Jin, Y., Zhao, S., et al. (2021). A review of model order reduction methods for large-scale structure systems. *Shock and Vibration*. https://doi.org/10.1155/2021/6631180

Martinez-Ruiz, A., & Lauro, N. C. (2023). Incremental singular value decomposition for some numerical aspects of multiblock redundancy analysis. *Computational Statistics*. https://doi.org/10.1007/s00180-023-01418-5

Matias, T., Souza, F., Araújo, R., Gonçalves, N., & Barreto, J. P. (2015). On-line sequential extreme learning machine based on recursive partial least squares. *Journal of Process Control, 27*, 15–21. https://doi.org/10.1016/j.jprocont.2015.01.004

Mifsud, M. J., MacManus, D. G., & Shaw, S. T. (2016). A variable-fidelity aerodynamic model using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids, 82*(10), 646–663. https://doi.org/10.1002/fld.4234

Mifsud, M., Zimmermann, R., & Görtz, S. (2015). Speeding-up the computation of high-lift aerodynamics using a residual-based reduced-order model. *CEAS Aeronautical Journal, 6*(1), 3–16. https://doi.org/10.1007/s13272-014-0125-0

Mu, H., He, F., Yuan, L., Commins, P., Wang, H., & Pan, Z. (2023). Toward a smart wire arc additive manufacturing system: A review on current developments and a framework of digital twin. *Journal of Manufacturing Systems, 67*, 174–189. https://doi.org/10.1016/j.jmsy.2023.01.012

Ou, G., Dyke, S. J., & Prakash, A. (2017). Real time hybrid simulation with online model updating: An analysis of accuracy. *Mechanical Systems and Signal Processing, 84*, 223–240. https://doi.org/10.1016/j.ymssp.2016.06.015

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks, 113*, 54–71. https://doi.org/10.1016/j.neunet.2019.01.012

Phalippou, P., Bouabdallah, S., Breitkopf, P., Villon, P., & Zarroug, M. (2020). 'On-the-fly' snapshots selection for proper orthogonal decomposition with application to nonlinear dynamics. *Computer Methods in Applied Mechanics and Engineering, 367*, 113120. https://doi.org/10.1016/j.cma.2020.113120

Phanden, R. K., Aditya, S. V., Sheokand, A., Goyal, K. K., Gahlot, P., & Jacso, A. (2022). A state-of-the-art review on implementation of digital twin in additive manufacturing to monitor and control parts quality. *Materials Today: Proceedings, 56*, 88–93. https://doi.org/10.1016/j.matpr.2021.12.217

Qian, W., Tang, M., Gao, H., Dong, J., Liang, J., & Liu, J. (2022). Improving indoor air flow and temperature prediction with local measurements based on CFD-EnKF data assimilation. *Building and Environment, 223*, 109511. https://doi.org/10.1016/j.buildenv.2022.109511

Ross, D. A., Lim, J., Lin, R. S., & Yang, M. H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision, 77*(1–3), 125–141. https://doi.org/10.1007/s11263-007-0075-7

Sajadi, P., Rahmani Dehaghani, M., Tang, Y., & Wang, G. G. (2024). Physics-informed online learning for temperature prediction in metal AM. *Materials, 17*(13), 3306. https://doi.org/10.3390/ma17133306

Sandmeyer Steel Company. (2016). Stainless steel plate: alloy 17–4PH. https://www.sandmeyersteel.com/17-4PH.html. Accessed 19 April 2024

Sarwar, S. S., Ankit, A., & Roy, K. (2020). Incremental learning in deep convolutional neural networks using partial network sharing. *IEEE Access, 8*, 4615–4628. https://doi.org/10.1109/ACCESS.2019.2963056

Saves, P., Lafage, R., Bartoli, N., Diouane, Y., Bussemaker, J., Lefebvre, T., et al. (2024). SMT 2.0: A surrogate modeling toolbox with a focus on hierarchical and mixed variables Gaussian processes. *Advances in Engineering Software, 188*, 103571. https://doi.org/10.1016/j.advengsoft.2023.103571

Shao, Y., Chen, J., Gu, X., Lu, J., & Du, S. (2024). A novel curved surface profile monitoring approach based on geometrical-spatial joint feature. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-024-02349-8

Su, S., Zhong, R. Y., Jiang, Y., Song, J., Fu, Y., & Cao, H. (2023). Digital twin and its potential applications in construction industry: State-of-art review and a conceptual framework. *Advanced Engineering Informatics, 57*, 102030. https://doi.org/10.1016/j.aei.2023.102030

Tang, Y., Rahmani Dehaghani, M., Sajadi, P., & Wang, G. G. (2024). Selecting subsets of source data for transfer learning with applications in metal additive manufacturing. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-024-02402-6

Vetrano, F., Mastroddi, F., & Ohayon, R. (2015). POD approach for unsteady aerodynamic model updating. *CEAS Aeronautical Journal, 6*(1), 121–136. https://doi.org/10.1007/s13272-014-0133-0

Xiang, F., Zhang, Z., Zuo, Y., & Tao, F. (2019). Digital twin driven green material optimal-selection towards sustainable manufacturing. *Procedia CIRP, 81*, 1290–1294. https://doi.org/10.1016/j.procir.2019.04.015

Yang, Q., Gu, Y., & Wu, D. (2019). Survey of incremental learning. In *Proceedings of the 31st Chinese Control and Decision Conference, CCDC 2019* (pp. 399–404). IEEE. https://doi.org/10.1109/CCDC.2019.8832774

Yavari, R., Smoqi, Z., Riensche, A., Bevans, B., Kobir, H., Mendoza, H., et al. (2021). Part-scale thermal simulation of laser powder bed fusion using graph theory: Effect of thermal history on porosity, microstructure evolution, and recoater crash. *Materials and Design, 204*, 109685. https://doi.org/10.1016/j.matdes.2021.109685

Yu, J., Song, Y., Tang, D., & Dai, J. (2021). A digital twin approach based on nonparametric Bayesian network for complex system health monitoring. *Journal of Manufacturing Systems, 58*, 293–304. https://doi.org/10.1016/j.jmsy.2020.07.005

Yuen, K. V., & Kuok, S. C. (2011). Bayesian methods for updating dynamic models. *Applied Mechanics Reviews, 64*(1), 010802. https://doi.org/10.1115/1.4004479

Zhang, L., Chen, X., Zhou, W., Cheng, T., Chen, L., Guo, Z., et al. (2020). Digital twins for additive manufacturing: A state-of-the-art review. *Applied Sciences (Switzerland), 10*(23), 1–10. https://doi.org/10.3390/app10238350

Zhu, X., & Ji, Y. (2023). A reduced order model based on adaptive proper orthogonal decomposition incorporated with modal coefficient learning for digital twin in process industry. *Journal of Manufacturing Processes, 102*, 780–794. https://doi.org/10.1016/j.jmapro.2023.07.061