East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.2024-210.060425

Vol. **16**, No. 1, pp. 204-226
February 2026

Improved Random Feature Method for Continuous Solution Reconstruction from Sparse Observation

Qingchun Li[1] and Qinwu Xu[2,*]

[1]*Nanjing University of Science and Technology ZiJin College, Nanjing 210023, China.*
[2]*School of Mathematics, Nanjing University, Nanjing 210000, China.*

Received 17 December 2024; Accepted (in revised version) 6 April 2025.

Abstract. The reconstruction of continuous solutions using all available mechanisms and data is essential for high-precision simulations and forecasts. This paper presents an improved random feature method (IRFM) that combines observational data with models for solution reconstruction. For the multi-region neuron approximation, we employ a global activation function with robust local approximation capabilities instead of the traditional piecewise method. This enhances smoothness across regions and accelerates convergence. We also derived the equivalence condition for the optimal solution of multi-constrained optimization problems, established criteria for determining the weights in the cost function and the distribution of randomly generated collocation points, reducing biases from subjective choices. Additionally, we introduce a weighted scheme for computing the cost function related to sparse observations, reducing interpolation errors and improving stability against noise. Numerical examples demonstrate that the IRFM is more accurate and converges faster than the original RFM. Its efficiency and accuracy are validated through comparisons with physics-informed neural networks, and its flexibility is shown by successful continuous solution reconstruction in complex domains.

AMS subject classifications: 65M32, 68T09

Key words: Improved random feature method, continuous solution reconstruction, data assimilation, sparse observation.

1. Introduction

Partial differential equation (PDE) models are widely used to describe the physical behavior of systems in various scientific and engineering applications. However, these models often suffer from inaccuracies due to limitations in the model itself, missing initial and boundary conditions, and parameter uncertainties, all of which affect the simulation's accuracy. By combining observational data with model predictions, it is possible to correct the model state effectively, significantly improving simulation precision. This is particularly

[*]Corresponding author. *Email address:* `xuqinwu@nju.edu.cn` (Q. Xu)

crucial in numerical forecasting, as it provides enhanced predictive accuracy when dealing with imperfect models and incomplete observational data.

To address uncertainties and sparse observational data, the problem can be formulated as the following optimization problem:

$$u^* = \arg\min_u \frac{\lambda}{2}\|u-u_0\|^2_{L^2(\Omega)} + \frac{1}{2}\|\mathscr{H}u-h\|^2_{L^2(\mathscr{D})}$$

subject to the PDE constraints

$$\mathscr{L}u(x,t) = f(x,t), \quad x\in\Omega, \quad t\in\mathscr{T},$$

where u is the continuous function to be reconstructed, $\mathscr{H}$ is the observation operator, h is the observation data, λ is the regularization parameter, and u_0 is the prior solution. The space domain is denoted as Ω, the time domain as $\mathscr{T}$, and the space-time domain as $\mathscr{D}=\Omega\times\mathscr{T}$.

When observational data are available at only a limited number of spatiotemporal points, the problem becomes one with sparse observations

$$u^* = \arg\min_u \frac{\lambda}{2}\|u-u_0\|^2_{L^2(\Omega)} + \frac{\Delta_x\Delta_t}{2}\sum_{j=1}^{N_t}\sum_{k=1}^{N_s}\big(\mathscr{H}u_{j,k}-h_{j,k}\big)^2,$$

where Δ_x and Δ_t are constants related to the density of spatial and temporal observation points, and $u_{j,k}$ and $h_{j,k}$ represent the solution and observations at discrete time and space points, respectively.

This type of problems are known as data assimilation in numerical forecasting [1]. The concept of data assimilation originated from meteorology [7] to integrate simulation results from weather models with real observations, improving the accuracy of weather forecasts. By the late 20th century, data assimilation had expanded to various fields, including Earth sciences [13], oceanography [17], and hydrology [15]. In these fields, we frequently encounter challenges related to imperfect models, incomplete observations, and data errors. Data assimilation offers an effective method to overcome these challenges by merging observational data with numerical models.

The reconstruction of continuous solutions from sparse observational data is one of the key challenges in data assimilation. In many applications, observational data is often limited and discontinuous, while PDE models require complete solutions across all spatial and temporal domains. For example, in numerical weather prediction (NWP), data may come from only a few weather stations or satellite paths, yet a comprehensive simulation of the atmospheric state demands a continuous solution in three-dimensional space and time [12]. This highlights the necessity of reconstructing high-precision continuous solutions from sparse observational data.

The primary methods of data assimilation include variational data assimilation (3D/4DVar) [20] and ensemble Kalman filter (EnKF) [10, 11], each with its own advantages and disadvantages. 3D/4DVar optimizes a cost function to find the best estimate of the system state, making it effective for integrating continuous observational data. However, it has high computational costs due to the repeated solving of state and adjoint equations, particularly with PDEs that have singular sources [8]. Recently, Cheng *et al.* [4] introduced VIVID, a new variational data assimilation scheme that uses deep learning inverse operators to manage sparse, unstructured sensor data, significantly improving efficiency through Voronoi tessellation and convolutional neural networks. In contrast, EnKF updates the state and covariance to estimate the optimal solution, making it suitable for real-time updates. However, it struggles with sparse observational data and faces challenges in nonlinear problem handling.

In recent years, with the advancement of machine learning and neural network technologies, data-driven approaches have opened new pathways for reconstructing continuous solutions from sparse observational data. Physics-informed neural network (PINNs) — cf. [18], incorporate physical equation constraints into the neural network's loss function, enabling efficient handling of nonlinear and high-dimensional problems while significantly reducing computational and storage demands. This method has been successfully applied across various fields, including ocean wave prediction [6], weather forecasting [5], and subsurface fluid transport [9]. To further improve reconstruction efficiency in complex systems, an echo state network (ESN) based on a modified PINN approach has been proposed, which transforms nonlinear dynamical systems with sparse observational data into a fixed-point problem [22]. Additionally, another study has demonstrated a flow field reconstruction method using PINNs, showcasing high-precision capabilities even with sparse or incomplete data [21]. These studies highlight the significant potential of PINNs for reconstructing continuous solutions in complex systems.

Despite the impressive performance of PINNs in reconstructing continuous solutions for complex systems, several limitations persist. First, the error analysis theory for neural networks is still underdeveloped, making precise error control difficult. Second, there are no clear guidelines for setting the combination weights of different constraints, such as those for dynamical systems, initial conditions, and boundary conditions, nor for determining the density and distribution of sampling points. Finally, the network requires retraining with each new set of observational data, which is time-consuming and make it unfriendly for real-time data assimilation.

This paper builds on the random feature method (RFM) introduced in the past two years [2, 3]. The RFM method uses a neuron-based basis function for function approximation, maintaining the flexibility of neural networks in approximating functions over complex domains. It converts optimization problems into systems of equations, thus avoiding the time-consuming training typical of neural networks while achieving high level of precision akin to high order finite element methods. However, the selection of weights for different components of the cost function remains subjective. Ensuring cross-region continuity of the solution requires a piecewise smooth partition of unity function, which hinders the global smoothness and computational efficiency of the solution. This paper establishes a flexible and highly accurate approximation for functions over complex domains based on RFM, derives the conditions for achieving the optimal solution of the objective function, provides guidelines for the weights of different components, and introduces a global extending strategy to replace the conventional partition of unity function in the multi-region

RFM method, enhancing training efficiency. The improved random feature method effectively reconstructs continuous solutions from sparse observational data, achieving superior accuracy and computational efficiency.

The remainder of this paper is organized as follows. Section 2 provides a detailed derivation of the algorithm, covering the formulation of the optimization model, neuron-based multi-region RFM approximation, an improved RFM, the derivation of equivalent conditions for the optimal solution, and the handling of sparse observations. Section 3 demonstrates the algorithm's accuracy and efficiency through various numerical experiments, including continuous solution reconstruction with RFM, the effectiveness of IRFM, comparisons with neural network-based methods, and reconstruction over complex domains. Finally, Section 4 concludes with a summary of the study's findings and contributions.

2. Reconstruction Algorithm Based on the Random Feature Method

To solve the optimization problem constrained by partial differential equations, we employ the Lagrange multiplier method to incorporate the deviations of the dynamical system and potential boundary condition deviations into the objective function, resulting in the following expression:

$$J[u]=\frac{\lambda_1}{2}\|u-u_0\|^2_{L^2(\Omega)}+\frac{\lambda_2}{2}\|\mathscr{H}u-h\|^2_{L^2(\mathscr{D})}+\frac{\lambda_3}{2}\|\mathscr{L}u-f\|^2_{L^2(\mathscr{D})}+\frac{\lambda_4}{2}\|u-g\|^2_{L^2(\partial\Omega\times\mathscr{T})}.$$

The continuous reconstructed solution $u^*(x,t)$ can be obtained by solving the unconstrained optimization problem that minimizes $J[u]$, viz.

$$u^*=\arg\min_u J[u]. \tag{2.1}$$

Here, $\lambda_1,\lambda_2,\lambda_3,\lambda_4$ are weight coefficients that can be determined in applications such as NWP using the empirical errors of the initial field and observational data. These coefficients reflect the relative importance of the errors of each term [14].

In data assimilation, the model state is updated by minimizing the weighted differences between the background field, observational data, and model forecasts. The background field typically comes from the previous model prediction or a statistical model and serves as an initial estimate of the system state, helping to blend or correct against observational data. It acts as a balancing component in the minimization process, preventing the model state from relying too heavily on observational data, especially when such data is sparse or erroneous. To regularize the model, the background field is often introduced as an initial-boundary condition.

2.1. Approximation method based on random features

To construct a flexible and efficient approximation method for the function $u(\boldsymbol{x},t)$ and solve the optimization problem, we introduce the random feature functions defined in the literature [2,3] to develop a multi-region random feature approximation method. Let the dimension of the spatial domain be d_x. We unify the treatment of the spatiotemporal domain, denoting $\boldsymbol{y}=(\boldsymbol{x},t)\in\mathscr{D}=\Omega\times[0,T]$, which results in the dimension of the spatiotemporal domain $\mathscr{D}$ being $d_y=d_x+1$. We partition the spatiotemporal domain $\mathscr{D}$ into multiple subregions, with the number of regions denoted as N_d, the i-th subregion is denoted as $\mathscr{D}_i$.

For each sub-region, we randomly generate a group of neurons, referred to as random feature function, which can be represented as

$$\phi_{i,j}(\boldsymbol{x},t)=\sigma\big(\boldsymbol{k}_{i,j}l_i(\boldsymbol{x},t)+b_{i,j}\big),\quad i=1,2,\dots,N_d,\quad j=1,2,\dots,N_i.$$

Once the network is initialized, the random feature functions are fixed. Using these functions, we construct a three-layer neural network for each sub-region, as shown in Fig. 1.

Let the number of neurons in the hidden layer of the i-th region's neural network be N_i. The corresponding local network for the sub-region $\mathscr{D}_i$ can be expressed as

$$u_h(\boldsymbol{x},t)|_{\mathscr{D}_i}=\sum_{j=1}^{N_i}\hat{u}_{i,j}\phi_{i,j}(\boldsymbol{x},t),\quad i=1,2,\dots,N_d. \tag{2.2}$$

In comparison to numerical methods like finite element method (FEM) [23], each neuron in the hidden layer acts as a basis function, termed here as a random feature function. The nonlinear activation function σ is typically chosen as the hyperbolic tangent function tanh or a trigonometric function, such as sin or cos. The symbol i denotes the index of the spatiotemporal subregion, $\boldsymbol{k}_{i,j}$ is the weight matrix for the spatiotemporal input, and $b_{i,j}$ is the bias term. These are independently and identically distributed, sampled uniformly as $\boldsymbol{k}_{i,j}\sim\mathbb{U}([-R_{ij},R_{ij}]^{d_y})$ and $b_{i,j}\sim\mathbb{U}([-R_{ij},R_{ij}])$. Once initialized randomly, they remain fixed. The choice of R_{ij} must be determined based on the basis functions, ensuring that the basis functions have strong approximation capabilities in local regions while avoiding the issue of vanishing gradients. For example, with the hyperbolic tangent activation function, setting $R_{ij}=1$ clearly provides strong approximation capabilities on the interval $[-1,1]$ without experiencing vanishing gradients.

Although this approximation method does not impose any restrictions on the choice of domain partitioning, in practical applications, each dimension is often partitioned independently, and the Cartesian product is then used to define the regions. For example, when $d_y=2$, the region $\mathscr{D}_{(i\cdot j)}$ can be defined as $\mathscr{D}_{(i\cdot j)}=[y_{i,j},y_{i+1,j}]\times[y_{i,j},y_{i,j+1}]$.

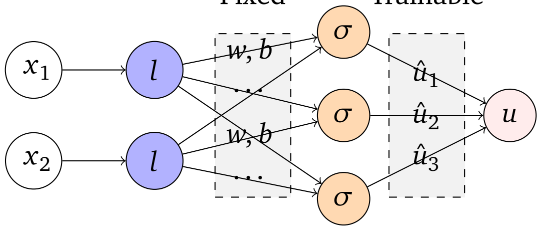


Figure 1: Schematic diagram of a neural network architecture on a single region.

The term $l_i(\boldsymbol{y}) = l_i(\boldsymbol{x},t)$ denotes a linear transformation applied to $(\boldsymbol{x},t)$, which maps $(\boldsymbol{x},t) \in \mathscr{D}_i = \Omega_{i_x} \times [t_{i_t-1}, t_{i_t}]$ onto $[-1,1]^{d_x+1}$. To define this mapping, we first compute the centers $\{\bar{\boldsymbol{y}}_n\}_{n=1}^{N_d}$ and radii $\{\boldsymbol{r}_n\}_{n=1}^{N_d}$ for all subregions as follows:

$$\begin{aligned}\bar{\boldsymbol{y}}_n &= (\bar{y}_{n,1}, \bar{y}_{n,2}, \cdots, \bar{y}_{n,d_y})^T \\ &= \left(\frac{y_{n,1\_\max} + y_{n,1\_\min}}{2}, \frac{y_{n,2\_\max} + y_{n,2\_\min}}{2}, \cdots, \frac{y_{n,d_y\_\max} + y_{n,d_y\_\min}}{2}\right)^T, \\ \boldsymbol{r}_n &= (r_{n,1}, r_{n,2}, \cdots, r_{n,d_y})^T \\ &= \left(\frac{y_{n,1\_\max} - y_{n,1\_\min}}{2}, \frac{y_{n,2\_\max} - y_{n,2\_\min}}{2}, \cdots, \frac{y_{n,d_y\_\max} - y_{n,d_y\_\min}}{2}\right)^T.\end{aligned}$$

Here, $y_{n,j\_\max} = \max_{\boldsymbol{y}\in\mathscr{D}_n}\{y_{n,j}\}$ and $y_{n,j\_\min} = \min_{\boldsymbol{y}\in\mathscr{D}_n}\{y_{n,j}\}$ represent the upper and lower bounds, respectively, of subregion $\mathscr{D}_n$ in the j-th dimension.

For each subregion $\mathscr{D}_n$, we define a linear transformation — viz.

$$l_n(\boldsymbol{y}) = \frac{1}{\boldsymbol{r}_n}(\boldsymbol{y} - \bar{\boldsymbol{y}}_n), \quad n = 1,2,\ldots,N_d, \quad \boldsymbol{y} \in \Omega \times [0,T].$$

This transformation resembles the input normalization in neural networks, mapping the local region $[\bar{y}_{n,1} - r_{n,1}, \bar{y}_{n,1} + r_{n,1}] \times [\bar{y}_{n,2} - r_{n,2}, \bar{y}_{n,2} + r_{n,2}] \times \cdots \times [\bar{y}_{n,d_y} - r_{n,d_y}, \bar{y}_{n,d_y} + r_{n,d_y}]$ onto the standard interval $[-1,1]^{d_y}$ to facilitate local feature fitting.

2.2. Approximation across different regions

When applied to a single region, the RFM method's performance does not improve significantly in accuracy, even with a substantial increase in the number of random feature functions. Constructing neural networks in segments is an effective way to enhance convergence speed. However, this approach requires addressing continuity issues at the boundaries between regions. Chen *et al.* [2] introduced the partition of unity (PoU) technique to ensure that functions maintain a certain level of continuity across region boundaries. With this technique, each subnetwork is combined to represent the model across the entire domain. Fig. 2 illustrates the combination of multiple local neural networks with the partition of unity function.

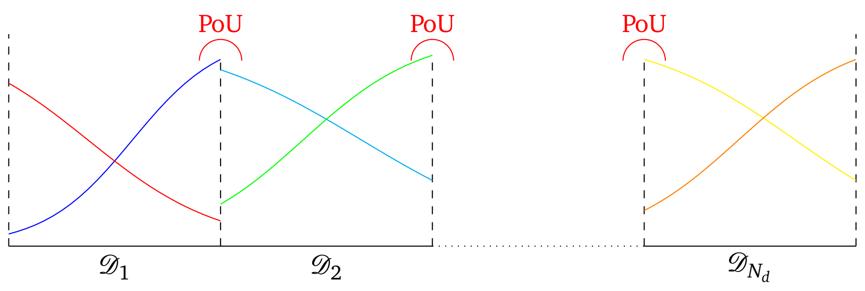


Figure 2: Local tanh basis for multi-region in 1D.

Let ψ_i, $1 \le i \le N_d$ be the partition of unity functions. By using this technique, we combine the output of each sub-region $\mathscr{D}_i$ in (2.2) to represent the model across the entire domain. This produces the following approximation of the function $u(\boldsymbol{y})$:

$$u(\boldsymbol{y}) \approx u_h(\boldsymbol{y}) = \sum_{i=1}^{N_d} \psi_i\big(l_i(\boldsymbol{y})\big) u_h(\boldsymbol{y})|_{\mathscr{D}_i} = \sum_{i=1}^{N_d} \psi_i\big(l_i(\boldsymbol{y})\big) \sum_{j=1}^{N_i} \hat{u}_{ij} \phi_{ij}(\boldsymbol{y}). \tag{2.3}$$

In [2], two commonly used PoU functions for one-dimensional case are presented — viz.

$$\text{PoU-1:} \quad \psi_n^a(x) = \mathbb{I}_{[-1,1]}(x),$$

$$\text{PoU-2:} \quad \psi_n^b(x) = \begin{cases} \dfrac{1+\sin(2\pi x)}{2}, & -\dfrac{5}{4} \le x < -\dfrac{3}{4}, \\ 1, & -\dfrac{3}{4} \le x < \dfrac{3}{4}, \\ \dfrac{1-\sin(2\pi x)}{2}, & \dfrac{3}{4} \le x < \dfrac{5}{4}, \\ 0, & \text{otherwise}. \end{cases}$$

However, the definition of PoU functions depends on a specific method of regional decomposition. In high-precision simulations of complex domain problems, unstructured adaptive decomposition is often necessary based on the domain's characteristics, resulting in complex partitioning that complicates the definition of the PoU function (see Fig. 3(a)). By employing global activation functions with strong local approximation capabilities, high-precision local approximations can be achieved without the PoU function, while ensuring smoothness across boundaries. Suitable functions for this purpose include hyperbolic tangent functions and radial basis functions. In this paper, we select the hyperbolic tangent function as the activation function.

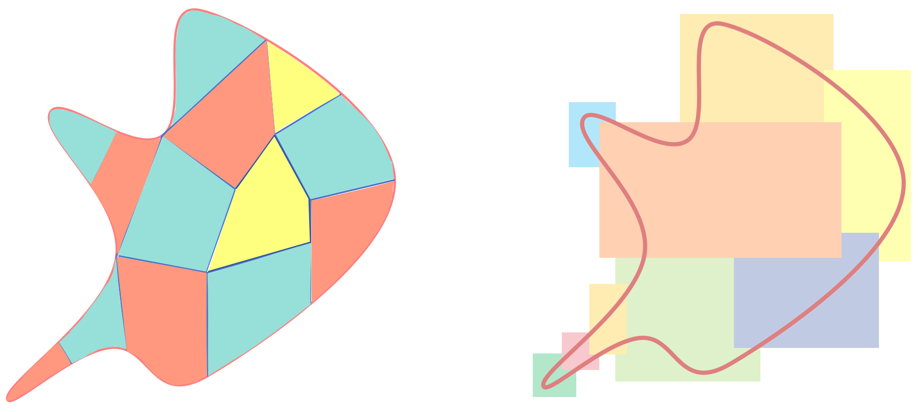

a) Unstructured decomposition b) Domain coverage through subregions

Figure 3: Schematic diagram of domain decomposition.

Instead of independently defining activation functions for each subregion as shown in Fig. 2, we extend the hyperbolic tangent activation function defined on each subregion to the global region, as depicted in Fig. 4.

After expanding the domain of the subregion activation functions, the Eq. (2.3) can be modified to obtain a simplified form of $u_h(\boldsymbol{y})$

$$u_h(\boldsymbol{y}) = \sum_{i=1}^{N_d}\sum_{j=1}^{N_i} \hat{u}_{ij}\phi_{ij}(\boldsymbol{y}), \quad \forall \boldsymbol{y} \in \mathscr{D}.$$

By definition, ϕ_{nj} is a globally continuously differentiable function, ensuring that $u_h(\boldsymbol{y})$ is also continuously differentiable. Thus, there is no need to address continuity and smoothness issues at the boundaries of subregions. The extended portions of the activation function close to constant, and their gradients tend toward zero, which does not diminish the approximation capability of the activation functions in other regions.

In addition, this improved RFM offers greater flexibility in sub-region partitioning. In the original RFM, the domain partitioning must be simple and non-overlapping to facilitate the definition of the PoU function. However, in the IRFM, due to the absence of constraints imposed by the partition of unity function, the partitioning rules for sub-regions can, in theory, be arbitrary, as long as they form a cover of the domain. Fig. 3(b) illustrates examples of rectangular coverings for complex regions.

In practice, the set of functions $\phi_{ij}(\boldsymbol{y})$, $i = 1,2,\ldots,N_d, j = 1,2,\ldots,N_i$ spans a linear space $\mathscr{S}$, defined as

$$\mathscr{S} = \left\{ u_h(\boldsymbol{y}) \in L^2(\Omega \times [0,T]) \,\middle|\, u_h(\boldsymbol{y}) = \sum_{i=1}^{N_d}\sum_{j=1}^{N_i} \hat{u}_{ij}\phi_{ij}(\boldsymbol{y}),\ \forall \hat{u}_{ij} \in \mathbb{R} \right\}.$$

The dynamical system described by linear PDE $\mathscr{L}u = f$ is solved in the linear space $\mathscr{S}$. Consequently, the discrete form of the dynamical system is simplified as

$$f(\boldsymbol{x},t) = \mathscr{L}u(\boldsymbol{x},t) \approx \mathscr{L}u_h(\boldsymbol{x},t) = \mathscr{L}\sum_{i=1}^{N_d}\sum_{j=1}^{N_i} \hat{u}_{ij}\phi_{ij}(\boldsymbol{x},t) = \sum_{i=1}^{N_d}\sum_{j=1}^{N_i} \hat{u}_{ij}\mathscr{L}\phi_{ij}(\boldsymbol{x},t).$$

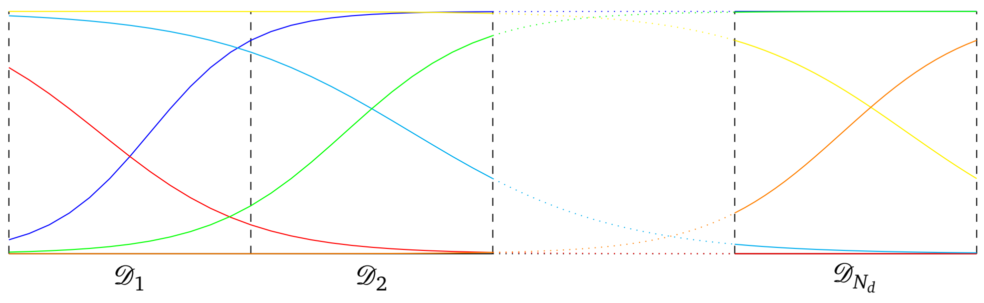


Figure 4: Global tanh basis for multi-region in 1D.

2.3. Solution of the unconstrained optimization problem

To obtain the optimal solution of the optimization problem (2.1), we need to search, amongst the space $\mathscr{S}$, of the solution that minimize $J[u]$. A general optimality condition is given by the variational inequality [14,16]. Assume u^* is the optimal solution in $\mathscr{S}$, then

$$\left(\nabla J[u^*], u-u^*\right) \geq 0 \quad \text{for all} \quad u \in \mathscr{S}.$$

Since for this quadratic optimization problem, $\mathscr{L}(u)$ and $\mathscr{H}(u)$ are linear functions, the cost functional $J[\phi]$ can be proved to convex, unique solution of the optimization problem is certifiable. In the linear space $\mathscr{S}$, variational inequality is reduced to the equality

$$\nabla J[u^*] = 0.$$

Assuming that the variable u is fully observable — i.e. for all $(\boldsymbol{x}, t) \in \Omega \times \mathscr{T}$, there exists $h(\boldsymbol{x}, t) = \mathscr{H}(u(\boldsymbol{x}, t))$. We consider the variation δJ,

$$\begin{aligned} \delta J = {} & \lambda_1\big(u-u_0, \delta u\big)_\Omega + \lambda_2\big(\mathscr{H}u-h, \mathscr{H}\delta u\big)_\mathscr{D} \\ & + \lambda_3\big(\mathscr{L}u-f, \mathscr{L}\delta u\big)_\mathscr{D} + \lambda_4\big(u-g, \delta u\big)_{\partial\Omega\times\mathscr{T}}. \end{aligned} \tag{2.4}$$

Unlike optimization problems in 3D and 4D variational methods in data assimilation, the inner product operations above are defined over three different regions $\Omega, \mathscr{D}$, and $\partial\Omega \times \mathscr{T}$. For computational consistency, we extend the domains of definition of u_0 and g to $\mathscr{D} = \Omega \times \mathscr{T}$, denoting the extended variables as u_0^E and g^E, and define them as follows:

$$u_0^E(\boldsymbol{x}, t) = \begin{cases} u_0(\boldsymbol{x}, 0), & (\boldsymbol{x}, t) \in \Omega \times [0, \Delta_t], \\ u(\boldsymbol{x}, t), & (\boldsymbol{x}, t) \in \Omega \times (\Delta_t, T], \end{cases}$$

$$g^E(\boldsymbol{x}, t) = \begin{cases} g(\boldsymbol{x}_\partial, t), & (\boldsymbol{x}, t) \in \partial_{\Delta_x}\Omega \times \mathscr{T}, \\ u(\boldsymbol{x}, t), & (\boldsymbol{x}, t) \in (\Omega \setminus \partial_{\Delta_x}\Omega) \times \mathscr{T}. \end{cases}$$

Here, Δ_t is a positive number much smaller than T, typically chosen as the average distance between adjacent sample points in the time domain for optimal approximation. After extending the definition of u_0, the inner product in the spatial region Ω can be extended to the spatiotemporal region $\mathscr{D}$, and the inner product of the initial term in $\mathscr{D}$ is not identically zero — i.e. $(u-u_0^E, \delta u)_\mathscr{D} = (u-u_0^E, \delta u)_{\Omega\times[0,\Delta_t]} \not\equiv 0$. Similarly, Δ_x is a small positive number, typically chosen as the average volume occupied by each spatial sample point. $\partial_{\Delta_x}\Omega$ denotes the boundary portion of the region Ω with width Δ_x, and $\boldsymbol{x}_\partial$ is the closest boundary point to $\boldsymbol{x}$. The boundary inner product is extended to $\mathscr{D}$, and $(u-g^E, \delta u)_\mathscr{D} \not\equiv 0$.

Thus, δJ can be uniformly written in the form

$$\delta J \approx \left(\frac{\lambda_1}{\Delta_t}\big(u-u_0^E\big) + \lambda_2\mathscr{H}^T(\mathscr{H}u-h) + \lambda_3\mathscr{L}^T(\mathscr{L}u-f) + \frac{\lambda_4}{\Delta_x}\big(u-g^E\big), \delta u\right)_\mathscr{D}, \tag{2.5}$$

where $\mathscr{H}^T, \mathscr{L}^T$ are the adjoint operators of $\mathscr{H}, \mathscr{L}$, respectively. Clearly, as $\Delta_t \to 0$ and $\Delta_x \to 0$, the expression (2.5) reduces to the exact form in (2.4).

The computation in the random feature method is performed through randomly generated collocation points. Let the number of random feature basis functions be N_u, so that $N_u = \sum_{i=1}^{N_d} N_i$. Since the number of random collocation points is usually much larger than the number of basis functions, the dimension of the space generated by all collocation points is much larger than that of the space $\mathscr{S}$. Therefore, it is unrealistic to enforce $\nabla J[u] = 0$ at all collocation points. We restrict u to the space $\mathscr{S}$, that is, assume u has the form given in (2.3), and set

$$\begin{aligned}
\hat{\boldsymbol{u}} &:= \big[\hat{u}_{1,1}, \cdots, \hat{u}_{1,N_1}, \hat{u}_{2,1}, \cdots, \hat{u}_{N_d,1}, \cdots, \hat{u}_{N_d,N_{N_d}}\big]^T, \\
\vec{\Phi}(\boldsymbol{y}) &:= \big[\Phi_{1,1}(\boldsymbol{y}), \cdots, \Phi_{1,N_1}(\boldsymbol{y}), \Phi_{2,1}, \cdots, \Phi_{N_d,1}, \cdots, \Phi_{N_d,N_{N_d}}\big].
\end{aligned}$$

Then, for any u, we have

$$u(\boldsymbol{y}) \approx u_h(\boldsymbol{y}) = \vec{\Phi}(\boldsymbol{y})\hat{\boldsymbol{u}}.$$

In this method, the calculation of inner products needs to be performed at random collocation points. Due to the differences between the spatial and temporal domains, we randomly select collocation points separately from the spatial and temporal domains, and then use the Cartesian product to obtain points within the region. Let the number of collocation points in the time domain be N_t and in the spatial domain be N_s, so that the total number of collocation points is $N_c = N_t \times N_s$. Each point in the region has an equal probability of being selected, with the average interval between adjacent sample points in the time domain denoted by Δ_t, and the average spatial region occupied by each sample point in the spatial domain denoted by Δ_x, consistent with the notation used in (2.5). Consequently, δJ can be written as

$$\begin{aligned}
\delta J \approx &\bigg(\frac{\lambda_1}{\Delta_t}\big(\vec{\Phi}(\boldsymbol{y})\hat{\boldsymbol{u}} - u_0^E\big) + \lambda_2 \mathscr{H}^T\big(\mathscr{H}\vec{\Phi}(\boldsymbol{y})\hat{\boldsymbol{u}} - h\big) \\
&\quad + \lambda_3 \mathscr{L}^T\big(\mathscr{L}\vec{\Phi}(\boldsymbol{y})\hat{\boldsymbol{u}} - f\big) + \frac{\lambda_4}{\Delta_x}\big(\vec{\Phi}(\boldsymbol{y})\hat{\boldsymbol{u}} - g^E\big),\ \delta\vec{\Phi}(\boldsymbol{y})\hat{\boldsymbol{u}} \bigg)_{\mathscr{D}} \\
\approx &\sum_{i=1}^{N_t}\sum_{j=1}^{N_s} \delta\hat{\boldsymbol{u}}^T \vec{\Phi}^T(\boldsymbol{x}_j, t_i)\Delta_t\Delta_x \\
&\times \bigg(\frac{\lambda_1}{\Delta_t}\big(\vec{\Phi}(\boldsymbol{x}_j, t_i)\hat{\boldsymbol{u}} - u_0^E(\boldsymbol{x}_j, t_i)\big) + \lambda_2 \mathscr{H}^T\big(\mathscr{H}\vec{\Phi}(\boldsymbol{x}_j, t_i)\hat{\boldsymbol{u}} - h(\boldsymbol{x}_j, t_i)\big) \\
&\quad + \lambda_3 \mathscr{L}^T\big(\mathscr{L}\vec{\Phi}(\boldsymbol{x}_j, t_i)\hat{\boldsymbol{u}} - f(\boldsymbol{x}_j, t_i)\big) + \frac{\lambda_4}{\Delta_x}\big(\vec{\Phi}(\boldsymbol{x}_j, t_i)\hat{\boldsymbol{u}} - g^E(\boldsymbol{x}_j, t_i)\big)\bigg) \\
= &\bigg(\sum_{i=1}^{N_t}\sum_{j=1}^{N_s} \vec{\Phi}^T(\boldsymbol{x}_j, t_i)\Delta_t\Delta_x \\
&\times \bigg(\frac{\lambda_1}{\Delta_t}\big(\vec{\Phi}(\boldsymbol{x}_j, t_i)\hat{\boldsymbol{u}} - u_0^E(\boldsymbol{x}_j, t_i)\big) + \lambda_2 \mathscr{H}^T\big(\mathscr{H}\vec{\Phi}(\boldsymbol{x}_j, t_i)\hat{\boldsymbol{u}} - h(\boldsymbol{x}_j, t_i)\big) \\
&\quad + \lambda_3 \mathscr{L}^T\big(\mathscr{L}\vec{\Phi}(\boldsymbol{x}_j, t_i)\hat{\boldsymbol{u}} - f(\boldsymbol{x}_j, t_i)\big) + \frac{\lambda_4}{\Delta_x}\big(\vec{\Phi}(\boldsymbol{x}_j, t_i)\hat{\boldsymbol{u}} - g^E(\boldsymbol{x}_j, t_i)\big)\bigg), \delta\hat{\boldsymbol{u}} \bigg). \qquad (2.6)
\end{aligned}$$

We arrange the collocation points in sequence, denote them as $\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_{N_c}$, and define the matrix

$$\boldsymbol{\Phi} = \begin{bmatrix} \Phi_{1,1}(\boldsymbol{y}_1) & \cdots & \Phi_{1,N_1}(\boldsymbol{y}_1) & \cdots & \Phi_{N_d,1}(\boldsymbol{y}_1) & \cdots & \Phi_{N_d,N_{N_d}}(\boldsymbol{y}_1) \\ \Phi_{1,1}(\boldsymbol{y}_2) & \cdots & \Phi_{1,N_1}(\boldsymbol{y}_2) & \cdots & \Phi_{N_d,1}(\boldsymbol{y}_2) & \cdots & \Phi_{N_d,N_{N_d}}(\boldsymbol{y}_2) \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \Phi_{1,1}(\boldsymbol{y}_{N_c}) & \cdots & \Phi_{1,N_1}(\boldsymbol{y}_{N_c}) & \cdots & \Phi_{N_d,1}(\boldsymbol{y}_{N_c}) & \cdots & \Phi_{N_d,N_{N_d}}(\boldsymbol{y}_{N_c}) \end{bmatrix}_{N_c \times N_u}.$$

Let $\mathbf{H}$ and $\mathbf{L}$ be the matrix representations of the operators $\mathscr{H}$ and $\mathscr{L}$, respectively, with their corresponding adjoint operators denoted as $\mathbf{H}^T$ and $\mathbf{L}^T$. We define the vectors

$$\begin{aligned} \mathbf{u}_0^E &= \left[u_0^E(\boldsymbol{y}_1), u_0^E(\boldsymbol{y}_2), \cdots, u_0^E(\boldsymbol{y}_{N_c})\right], \quad \mathbf{h} = \left[h(\boldsymbol{y}_1), h(\boldsymbol{y}_2), \cdots, h(\boldsymbol{y}_{N_c})\right], \\ \mathbf{g}^E &= \left[g^E(\boldsymbol{y}_1), g^E(\boldsymbol{y}_2), \cdots, g^E(\boldsymbol{y}_{N_c})\right], \quad \mathbf{f} = \left[f(\boldsymbol{y}_1), f(\boldsymbol{y}_2), \cdots, f(\boldsymbol{y}_{N_c})\right]. \end{aligned}$$

To satisfy the optimality condition, the left-hand side of the inner product in (2.6) should identically equal $\mathbf{0}$. By substituting the discrete operators, configuration matrices, and vectors into (2.6), we obtain that the optimal value $\hat{\boldsymbol{u}}^*$ satisfies

$$\begin{aligned} &\frac{\lambda_1}{\Delta_t}\boldsymbol{\Phi}^T\left(\boldsymbol{\Phi}\hat{\boldsymbol{u}}^* - \mathbf{u}_0^E\right) + \lambda_2 \boldsymbol{\Phi}^T \mathbf{H}^T (\mathbf{H}\boldsymbol{\Phi}\hat{\boldsymbol{u}}^* - \mathbf{h}) \\ &\qquad + \lambda_3 \boldsymbol{\Phi}^T \mathbf{L}^T (\mathbf{L}\boldsymbol{\Phi}\hat{\boldsymbol{u}}^* - \mathbf{f}) + \frac{\lambda_4}{\Delta_x}\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\hat{\boldsymbol{u}}^* - \mathbf{g}^E) = \mathbf{0}. \end{aligned} \tag{2.7}$$

Rearranging the Eqs. (2.7) and introducing the notation

$$\begin{aligned} \mathbf{A} &= \boldsymbol{\Phi}^T\left(\frac{\lambda_1}{\Delta_t}\boldsymbol{\Phi} + \lambda_2 \mathbf{H}^T\mathbf{H}\boldsymbol{\Phi} + \lambda_3 \mathbf{L}^T\mathbf{L}\boldsymbol{\Phi} + \frac{\lambda_4}{\Delta_x}\boldsymbol{\Phi}\right), \\ \mathbf{b} &= \frac{\lambda_1}{\Delta_t}\boldsymbol{\Phi}^T\mathbf{u}_0^E + \lambda_2\boldsymbol{\Phi}^T\mathbf{H}^T\mathbf{h} + \lambda_3\boldsymbol{\Phi}^T\mathbf{L}^T\mathbf{f} + \frac{\lambda_4}{\Delta_x}\boldsymbol{\Phi}^T\mathbf{g}^E, \end{aligned}$$

we obtain the linear system $\mathbf{A}\hat{\boldsymbol{u}}^* = \mathbf{b}$. It is evident that with a sufficient number of collocation points, we have $\operatorname{rank}(\boldsymbol{\Phi}) \geq N_u$, ensuring that $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ is an invertible matrix. Similarly, we can deduce that $\mathbf{A}$ is an invertible $N_u \times N_u$ matrix, and the linear system $\mathbf{A}\hat{\boldsymbol{u}}^* = \mathbf{b}$ has a unique solution.

The presence of Δ_t and Δ_x in the equation is essential to balance the number of collocation points and their weight across different regions. The number of collocation points in the initial value domain $\Omega \times [0]$ varies only with the density of spatial points. As the density of temporal samples increases, the number of collocation points associated with the dynamic system constraint also increases, while the initial condition points remain unchanged. In this case, the initial condition weight λ_1/Δ_t increases, offsetting the increased emphasis from additional dynamic system collocation points. Similarly, the boundary term weight λ_4/Δ_x plays a similar role in balancing weights.

2.4. Strategy for noisy sparse observation problems

The main differences between the noisy sparse observation problem and the precise fully observed problem are:

1. The observed value function $h(\boldsymbol{x},t)$ is defined only at a few discrete points, with observation points often misaligned with the random collocation points.
2. The observed values are inaccurate, and over-reliance on individual observations can introduce errors into the reconstruction solution.

As a result, standard integration using collocation points is not applicable. We need a viable approach for calculating the following term related to sparse observations:

$$V_H = \sum_{i=1}^{N_t}\sum_{j=1}^{N_s} \vec{\Phi}^T(\boldsymbol{x}_j, t_i)\Delta_t\Delta_x\lambda_2\mathscr{H}^T\big(\mathscr{H}\vec{\Phi}(\boldsymbol{x}_j, t_i)\hat{\boldsymbol{u}} - h(\boldsymbol{x}_j, t_i)\big).$$

A straightforward approach is interpolation, which uses sparse observation points as interpolation nodes to construct the observed function over the entire region and thus obtain function values at the collocation points. However, when observation points are too sparse, this can produce interpolated observation values that deviate significantly from the true solution. An alternative approach to avoid introducing artificial observational errors is to perform numerical integration directly based on the observation points instead of the collocation points. In this case, the weights associated with the density of integration points are adjusted according to the average spatial and temporal coverage of each observation point, similar to how initial and boundary conditions are treated.

Let the number of observation points be N_o, and denote the set of observation points as $\{\boldsymbol{y}_1^{obs}, \boldsymbol{y}_2^{obs}, \cdots, \boldsymbol{y}_{N_o}^{obs}\}$, the observed values $h_i^{obs} = u(\boldsymbol{y}_i^{obs})$. We then have

$$V_H = \frac{N_c}{N_o}\sum_{i=1}^{N_o} \vec{\Phi}^T(\boldsymbol{y}_i^{obs})\Delta_t\Delta_x\lambda_2\mathscr{H}^T(\mathscr{H}\vec{\Phi}(\boldsymbol{y}_i^{obs})\hat{\boldsymbol{u}} - h_i^{obs}).$$

When $N_c \gg N_o$, individual observation points receive excessively high weights, leading to the introduction of observation errors from noisy data into the solution. In fields like geography, it is often stated that “all attribute values on a geographic surface are related to each other, but closer values are more strongly related than are more distant ones [19]”. Therefore, treating the observed values as a weighted average of nearby collocation points, rather than as precise observations at corresponding positions, is a reasonable strategy to mitigate over-reliance on noisy data. For the weighting function, the Gaussian function $\exp(-\|\boldsymbol{y}_j - \boldsymbol{y}_i^{obs}\|^2/\sigma^2)$ is a popular choice. In implementation, we redefine the observation matrix $\mathbf{W}_{N_o\times N_c}$ with elements

$$\mathbf{W}_{i,j} = \frac{1}{\alpha_i}\exp\left(-\frac{\|\boldsymbol{y}_j - \boldsymbol{y}_i^{obs}\|^2}{\sigma^2}\right),$$

where α_i is a scaling factor that ensures $\sum_{j=1}^{N_c} \mathbf{W}_{i,j} = 1$. With this, we obtain the following system of linear equations:

$$\begin{aligned}&\mathbf{\Phi}^T\left(\frac{\lambda_1}{\Delta_t}\mathbf{\Phi} + \frac{\lambda_2 N_c}{N_o}\mathbf{H}^T\mathbf{W}^T\mathbf{W}\mathbf{H}\mathbf{\Phi} + \lambda_3\mathbf{L}^T\mathbf{L}\mathbf{\Phi} + \frac{\lambda_4}{\Delta_x}\mathbf{\Phi}\right)\hat{\boldsymbol{u}}^* \\ &= \frac{\lambda_1}{\Delta_t}\mathbf{\Phi}^T\mathbf{u}_0^E + \frac{\lambda_2 N_c}{N_o}\mathbf{\Phi}^T\mathbf{H}^T\mathbf{W}^T\mathbf{h}_{obs} + \lambda_3\mathbf{\Phi}^T\mathbf{L}^T\mathbf{f} + \frac{\lambda_4}{\Delta_x}\mathbf{\Phi}^T\mathbf{g}^E.\end{aligned}$$

Through the above derivation, the solution of reconstruction problem based on the RFM has been converted into solving a system of linear equations, avoiding the highly time-consuming neural network training process. Compared to the finite element method, this approach circumvents grid partitioning and stiffness matrix assembly by leveraging the randomness of feature functions and collocation points while retaining the neural network's capacity and flexibility to approximate complex regions. Solving this system yields the optimal solution $\hat{\boldsymbol{u}}^*$, and the analytical value at any collocation point $\boldsymbol{y} = (\boldsymbol{x}, t)$ within the domain $\Omega \times [0, T]$ is given by

$$u^{\mathrm{a}}(\boldsymbol{x}, t) = \left[\Phi_{1,1}(\boldsymbol{y}), \cdots, \Phi_{1,N_1}(\boldsymbol{y}), \cdots, \Phi_{N_d,1}(\boldsymbol{y}), \cdots, \Phi_{N_d,N_{N_d}}(\boldsymbol{y})\right]\hat{\boldsymbol{u}}^*.$$

Algorithm 2.1 shows detailed steps for continuous solution reconstruction using the improved random feature method.

Algorithm 2.1 IRFM for Continuous Solution Reconstruction

```
1: **Input:** Spatiotemporal domain $\Omega \times [0, T]$, initial values, boundary conditions, collocation points $\{\boldsymbol{y}_n\}_{n=1}^{N_c}$, observation points $\{\boldsymbol{y}_n^{obs}\}_{n=1}^{N_o}$, observation values $\{h_n^{obs}\}_{n=1}^{N_o}$.
2: Partition the spatiotemporal domain $\Omega \times [0, T]$ into $N_d$ subregions $\{\mathscr{D}_n\}_{n=1}^{N_d}$.
3: **for** each subregion $\mathscr{D}_n$ **do**
4:     Compute center $\bar{\boldsymbol{y}}_n$ and radius $\boldsymbol{r}_n$ for subregion $\mathscr{D}_n$.
5:     Construct linear transformation $l_n(\boldsymbol{y}) = (1/\boldsymbol{r}_n)(\boldsymbol{y} - \bar{\boldsymbol{y}}_n)$.
6:     Generate $N_n$ random feature functions
           $\phi_{nj}(\boldsymbol{y}) = \sigma\big(\boldsymbol{k}_{nj} l_n(\boldsymbol{y}) + b_{nj}\big), \quad j = 1, 2, \ldots, N_n.$
7: Generate the collocation points $\{\boldsymbol{y}_n\}_{n=1}^{N_c}$ and assemble the matrix $\mathbf{\Phi}$.
8: Calculate the average time interval $\Delta_t$ and average spatial volume per sample $\Delta_x$.
9: Assemble the observational matrix $\mathbf{H}$ and weight matrix $\mathbf{W}$.
10: Assemble the coefficients matrix $\mathbf{A}$ and the right hand side vector $\mathbf{b}$
           $\mathbf{A} = \mathbf{\Phi}^T\left(\frac{\lambda_1}{\Delta_t}\mathbf{\Phi} + \frac{\lambda_2 N_c}{N_o}\mathbf{H}^T\mathbf{W}^T\mathbf{W}\mathbf{H}\mathbf{\Phi} + \lambda_3\mathbf{L}^T\mathbf{L}\mathbf{\Phi} + \frac{\lambda_4}{\Delta_x}\mathbf{\Phi}\right),$
           $\mathbf{b} = \frac{\lambda_1}{\Delta_t}\mathbf{\Phi}^T\mathbf{u}_0^E + \frac{\lambda_2 N_c}{N_o}\mathbf{\Phi}^T\mathbf{H}^T\mathbf{W}^T\mathbf{h}_{obs} + \lambda_3\mathbf{\Phi}^T\mathbf{L}^T\mathbf{f} + \frac{\lambda_4}{\Delta_x}\mathbf{\Phi}^T\mathbf{g}^E.$
11: Solve the linear equations $\boldsymbol{A}\hat{\boldsymbol{u}}^* = \boldsymbol{b}$ to obtain the optimal coefficients $\hat{\boldsymbol{u}}^*$.
12: **Output:** Reconstructed solution $u^{\mathrm{a}}(\boldsymbol{y}) = \vec{\Phi}(\boldsymbol{y})\hat{\boldsymbol{u}}^*$.
```

3. Numerical Results

This section validates the accuracy and efficiency of IRFM through several numerical examples. We first evaluate its performance in solving static and time-dependent PDEs, then apply it to continuous solution reconstruction problems, comparing it with physics-informed neural networks. Finally, we implement continuous solution reconstruction using IRFM on a complex two-dimensional domain to demonstrate the method's flexibility. For all the examples presented, unless otherwise specified, we assume that the observational data and the model have equal reliability, with weights set to $\lambda_2 = \lambda_3 = 1$. The initial values and boundary conditions are rough estimates with limited accuracy, serving only as regularization terms. To prevent the rough guess from affecting the solution's accuracy, we assign it a very small weight of $\lambda_1 = \lambda_4 = 10^{-7}$.

3.1. IRFM for solution of PDEs

We use a one-dimensional Helmholtz equation to validate the accuracy of the improved RFM for the solving static PDE. The equation has the form

$$\begin{aligned}&\frac{d^2u(x)}{dx^2} - \omega u(x) = f(x), \quad x \in [0,8],\\&u(0) = c_1, \quad u(8) = c_2.\end{aligned}$$

We set $\omega = 4$, with the exact solution given by

$$u(x) = \sin\left(3\pi x + \frac{3\pi}{20}\right)\cos\left(2\pi x + \frac{\pi}{10}\right) + 2.$$

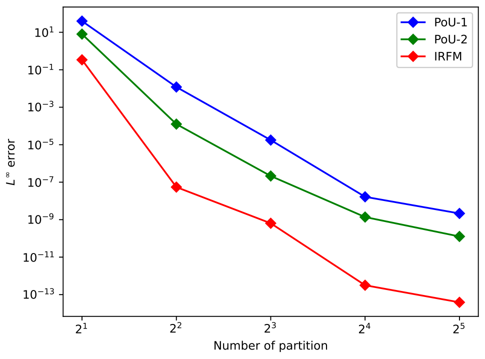
From this, c_1, c_2, and $f(x)$ can be computed as follows:

$$\begin{aligned}f(x) = &-13\pi^2 \sin\left(3\pi x + \frac{3\pi}{20}\right)\cos\left(2\pi x + \frac{\pi}{10}\right)\\&- 12\pi^2 \cos\left(3\pi x + \frac{3\pi}{20}\right)\sin\left(2\pi x + \frac{\pi}{10}\right),\\u(0) = u(8) = &\sin\frac{3\pi}{20}\cos\frac{\pi}{10} + 2.\end{aligned}$$

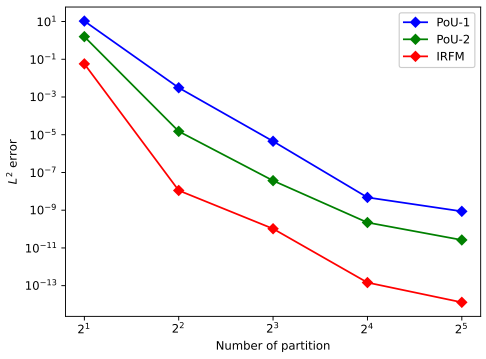
We solve the equation by three different versions of RFM (RFM with PoU-1, RFM with PoU-2, Improved RFM) with the following hyperparameter settings:

- Number of feature functions per subdomain: $N_i = 50$.
- Number of collocation points per subdomain: $Q_i = 50$.
- Number of subdomains: $N_d = 2, 4, 8, 16, 32$.

This subsection addresses the direct solution of the PDE problem with precise initial and boundary conditions but no observational data; therefore, the weights are set as $\lambda_1 = \lambda_3 =$



(a) L^∞ error



(b) L^2 error

Figure 5: Validation of the IRFM. The horizontal axis represents the number of subdomains, and the vertical axis represents error. Both axes use logarithmic scales.

Table 1: Comparison of three versions of RFM for the one-dimensional Helmholtz equation.

N_d	RFM with PoU-1		RFM with PoU-2		Improved RFM	
	L^∞ error	L^2 error	L^∞ error	L^2 error	L^∞ error	L^2 error
2	39.58	10.55	8.00	1.58	0.34	0.057
4	1.20E-2	3.09E-3	1.26E-4	1.51E-05	5.48E-08	1.11E-08
8	1.77E-05	4.50E-06	2.14E-07	3.68E-08	6.43E-10	1.04E-10
16	1.66E-08	4.75E-09	1.38E-09	2.22E-10	3.17E-13	1.46E-13
32	2.15E-09	8.69E-10	1.27E-10	2.64E-11	3.89E-14	1.31E-14

$\lambda_4 = 1$ and $\lambda_2 = 0$. The results, displayed in Fig. 5 and Table 1, show that the improved RFM converges faster than the RFM with PoU-1 and PoU-2 for solving the Helmholtz equation under the same hyperparameters, with L^∞ and L^2 errors reduced by several orders of magnitude. These findings underscore the effectiveness of the improved RFM for static PDEs.

Next, we use a one-dimensional diffusion equation to validate the performance of the improved RFM for solving dynamic PDE. The equation is given as follows:

$$\begin{aligned} &\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} = f(x,t), && (x,t) \in [0,5] \times [0,1], \\ &u(0,t) = g_1(t), && t \in [0,1], \\ &u(5,t) = g_2(t), && t \in [0,1], \\ &u(x,0) = h(x), && x \in [0,5]. \end{aligned}$$

We assume the exact solution to be

$$\begin{aligned} u(x,t) = &\left[2\cos\left(\pi x + \frac{\pi}{5}\right) + \frac{3}{2}\cos\left(2\pi x - \frac{3\pi}{5}\right)\right] \\ &\times \left[2\cos\left(\pi t + \frac{\pi}{5}\right) + \frac{3}{2}\cos\left(2\pi t - \frac{3\pi}{5}\right)\right]. \end{aligned}$$

Consequently, $f(x,t)$, $g_1(t)$, $g_2(t)$, and $h(x)$ have the form

$$
\begin{aligned}
f(x,t) = & \left[2\cos\left(\pi x + \frac{\pi}{5}\right) + \frac{3}{2}\cos\left(2\pi x - \frac{3\pi}{5}\right)\right] \\
& \times \left[-2\pi\sin\left(\pi t + \frac{\pi}{5}\right) - 3\pi\sin\left(2\pi t - \frac{3\pi}{5}\right)\right] \\
& + \nu\left[2\pi^2\cos\left(\pi x + \frac{\pi}{5}\right) + 6\pi^2\cos\left(2\pi x - \frac{3\pi}{5}\right)\right] \\
& \times \left[2\cos(\pi t + \frac{\pi}{5}) + \frac{3}{2}\cos\left(2\pi t - \frac{3\pi}{5}\right)\right], \\
g_1(t) = & \left[2\cos\frac{\pi}{5} + \frac{3}{2}\cos\frac{3\pi}{5}\right]\left[2\cos\left(\pi t + \frac{\pi}{5}\right) + \frac{3}{2}\cos\left(2\pi t - \frac{3\pi}{5}\right)\right], \\
g_2(t) = & \left[-2\cos\frac{\pi}{5} + \frac{3}{2}\cos\frac{3\pi}{5}\right]\left[2\cos\left(\pi t + \frac{\pi}{5}\right) + \frac{3}{2}\cos\left(2\pi t - \frac{3\pi}{5}\right)\right], \\
h(x) = & \left[2\cos\left(\pi x + \frac{\pi}{5}\right) + \frac{3}{2}\cos\left(2\pi x - \frac{3\pi}{5}\right)\right]\left[2\cos\frac{\pi}{5} + \frac{3}{2}\cos\frac{3\pi}{5}\right].
\end{aligned}
$$

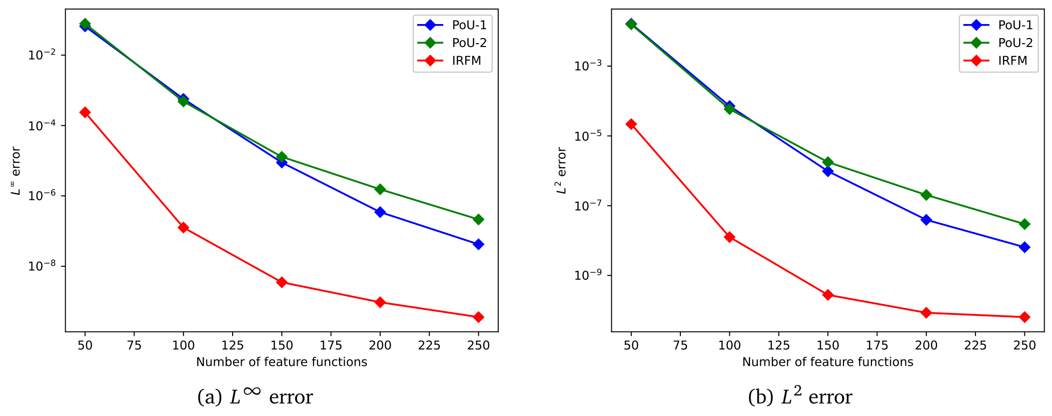
The diffusion coefficient is set to $\nu = 0.01$, and the hyperparameter settings are as follows:

- The number of partitions in the spatial dimension is $N_x = 5$.
- The number of partitions in the time dimension is $N_t = 2$.
- The number of collocation points in each subregion's spatial dimension is $Q_x = 30$.
- The number of collocation points in each subregion's time dimension is $Q_t = 30$.
- The number of feature functions in each subregion is set sequentially as $N_i = 50, 100, 150, 200, 250$.

Thus, the number of collocation points per sub-region is $Q_i = Q_x \times Q_t = 900$. The results, presented in Fig. 6 and Table 2, clearly show that the improved RFM converges significantly faster than those using the partition of unity function, achieving error levels several orders of magnitude lower under the same hyperparameters. This numerical example demonstrates that the IRFM significantly improves both convergence speed and accuracy compared to the RFM method using the PoU function. Although the IRFM generates a denser matrix compared to the original RFM, it eliminates the need to compute the PoU function, resulting in a computational cost comparable to that of the original RFM. The comparison results are shown in Table 3.

For the one-dimensional diffusion problem, we also employ a PINN to obtain the solution. To ensure a fair comparison between the performance of the IRFM and PINN, we conduct computations using identical collocation points. The PINN architecture consists of 4 hidden layers, each containing 32 neurons. The learning rate is initialized at 10^{-3} and follows an annealing schedule, reducing by half every 2,000 epochs, with training proceeding for a total of 20,000 epochs.

The numerical results obtained from the PINN exhibit an L^∞ error of 7.38×10^{-1} and an L^2 error of 1.89×10^{-2}. By comparing these results with the IRFM results presented in Table 2, it is evident that the accuracy of the PDE solution achieved by the PINN is significantly lower than that of both the RFM and IRFM. This highlights the superior performance of the IRFM in terms of numerical precision and robustness.



(a) L^∞ error (b) L^2 error

Figure 6: Effectiveness validation of the Improved RFM. The horizontal axis represents the number of feature functions in each subregion, and the vertical axis shows the error in logarithmic scale.

Table 2: Comparison of the three versions of RFM for the 1D diffusion equation.

N_i	RFM with PoU-1		RFM with PoU-2		Improved RFM	
	L^∞ error	L^2 error	L^∞ error	L^2 error	L^∞ error	L^2 error
50	6.60E-02	1.64E-02	7.82E-02	1.58E-02	2.37E-04	2.18E-05
100	5.70E-04	7.17E-05	4.83E-04	5.87E-05	1.26E-07	1.26E-08
150	8.89E-06	9.73E-07	1.29E-05	1.77E-06	3.51E-09	2.78E-10
200	3.46E-07	3.93E-08	1.54E-06	2.03E-07	9.48E-10	8.54E-11
250	4.21E-08	6.46E-09	2.14E-07	2.97E-08	3.59E-10	6.41E-11

Table 3: Training time (in seconds) of three RFM variants for the 1D diffusion equation.

N_i	RFM with PoU-1	RFM with PoU-2	Improved RFM
200	4.053	4.422	4.093
250	7.172	7.386	7.579
300	11.790	12.198	12.549
350	17.838	19.183	18.919
400	28.098	27.111	27.179

3.2. IRFM-based continuous solution reconstruction

We use the one-dimensional convection-diffusion equation (CDE) to perform continuous solution reconstruction. The original equation is as follows:

$$\frac{\partial u}{\partial t}+v\frac{\partial u}{\partial x}-v\frac{\partial^2 u}{\partial x^2}+u=f(x,t),\quad (x,t)\in[0,5]\times[0,1].$$

We set the diffusion coefficient $v=0.01$, convection velocity $v=0.01$, and assume the exact solution $u_e(x,t)=e^{-t}\sin(\pi(x+t+x\cdot t))$, from which $f(x,t)$ is derived. We select 10 observation points uniformly in space, denoted as $\hat{X}$. In the time domain, data is collected at intervals of $\Delta t=0.1$, denoted as $\hat{T}$. The total observation collocation points are formed by the Cartesian product of $\hat{X}$ and $\hat{T}$. The observation values are computed from the exact solution with the addition of Gaussian noise.

To formulate a regularized problem, we provide an initial guess for the initial and boundary conditions. In this section, we obtain this guess by adding a perturbation function generated by a Gaussian radial basis function to the exact initial and boundary conditions. The continuous solution reconstruction problem is presented as

$$\begin{aligned}
&\frac{\partial u}{\partial t}+v\frac{\partial u}{\partial x}-v\frac{\partial^2 u}{\partial x^2}+u=f(x,t), && (x,t)\in[0,5]\times[0,1],\\
&h(x_i,t_j)=u_e(x_i,t_j)+Noise, && (x_i,t_j)\in\{\hat{X}\times\hat{T}\},\\
&u(0,t)=u_e(0,t)+RBF(t), && t\in[0,1],\\
&u(5,t)=u_e(5,t)+RBF(t), && t\in[0,1],\\
&u(x,0)=u_e(x,0)+RBF(x), && x\in[0,5].
\end{aligned}\tag{3.1}$$

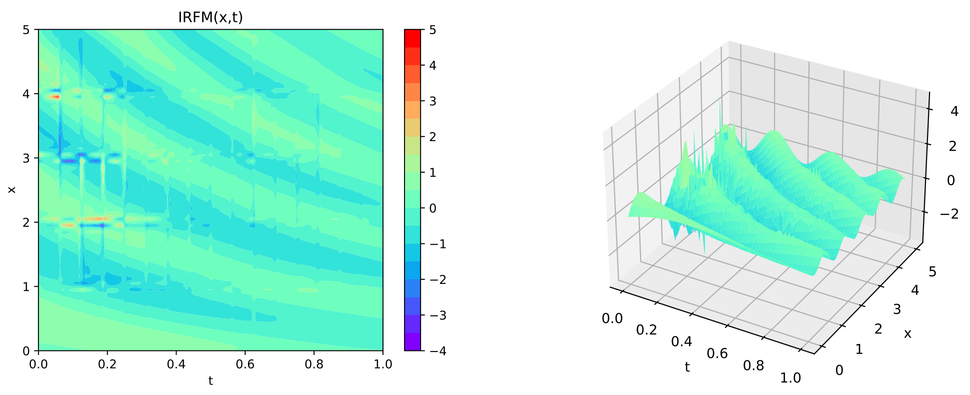
We solve this problem using IRFM, with the hyperparameters set as follows:

- Number of divisions for the spatial dimension: $N_x=5$.
- Number of collocation points in each spatial sub-region: $Q_x=10$.
- Number of collocation points in each temporal sub-region: $Q_t=10$.
- Number of feature functions in each sub-region: $N_i=100$.
- Number of divisions for the time dimension is set sequentially as $N_t=2,4,8,16$.

To assess the impact of observation data on the reconstructed solution, we solve this problem without observation constraints ($\lambda_2=0$) and with observation constraints ($\lambda_2=1$) separately. First with $\lambda_2=0$, we perform reconstruction based solely on the partial differential equations and the perturbed initial and boundary conditions. The results are shown in Table 4(a). The error remains stable at a fixed order, with the L^∞ error around 10^0 and the L^2 error around 10^{-1}. Fig. 7 shows the reconstructed solution with noise for $N_t=16$. The results are obviously unsatisfactory. Then we re-solved this problem with $\lambda_2=1$. The results, presented in Table 4(b), indicate a significant reduction in the error

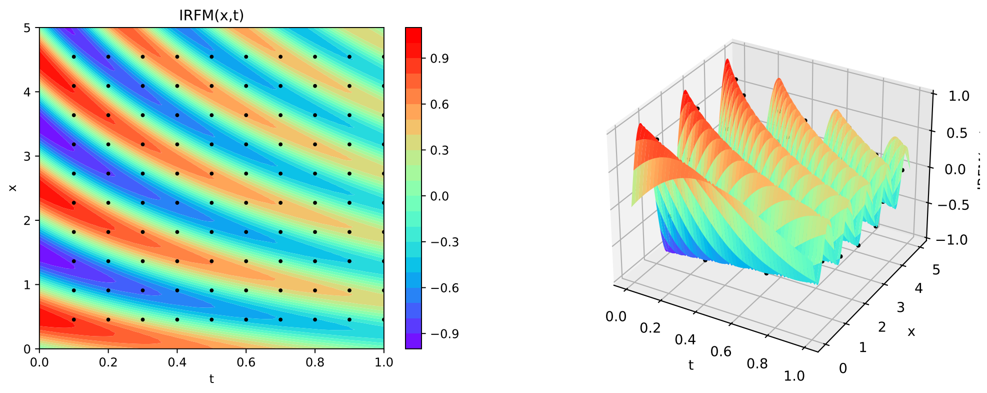
Table 4: Error of IRFM based reconstructed solution.

(a) without observation			(b) with observation		
N_t	L^∞ Error	L^2 Error	N_t	L^∞ Error	L^2 Error
2	1.23	1.64E-01	2	1.24E-01	2.50E-02
4	1.79	1.86E-01	4	6.40E-03	1.28E-03
8	2.03	2.08E-01	8	2.21E-03	4.43E-04
16	4.52	2.80E-01	16	5.95E-04	4.52E-05



(a) Contour plot of the numerical solution (b) 3D plot of the numerical solution

Figure 7: Solution for the 1D CDE with disturbed initial and boundary conditions.



(a) Contour of the analysis solution (b) 3D view of the analysis solution

Figure 8: IRFM based reconstructed solution for the 1D CDE.

between the model output and the true solution after introducing observations, leading the system's evolution to better align with the true state. Fig. 8 shows the reconstructed solution with observations when $N_t = 16$. The black dots in the figure represent the observation data. This demonstrates the advantage of the IRFM in continuous solution reconstruction.

3.3. Comparison with PINNs

For the one-dimensional convection-diffusion problem discussed in Section 3.2 and Eq. (3.1), we solve it using a PINN. To compare the performance of IRFM and PINN in continuous solution reconstruction, we perform computations on the same machine with identical collocation points and observational data. The neural network consists of 6 hidden layers, each with 32 neurons. The learning rate starts at 10^{-2} and follows an annealing schedule, halving every 5,000 epochs until reaching a total of 100,000 epochs.

The PINN results show a training time of 21 minutes and 33 seconds, with an L^∞ error of 1.16×10^{-1} and an L^2 error of 6.07×10^{-2}. Comparing these results with the IRFM results in Table 4(b), it is evident that the accuracy of reconstructed solution using PINN is significantly lower than that of the IRFM. Additionally, the IRFM demonstrates a clear advantage in computation time: with $N_t = 2$, it takes only 1.2 seconds, which is much less than the PINN's computation time.

3.4. Problems in complex geometric domains

Consider the membrane vibration equation (MVE)

$$\frac{\partial^2 u}{\partial t^2} - \mathbf{c} \cdot \Delta u = f(x,y,t), \quad (x,y,t) \in \Omega \times [0,T],$$

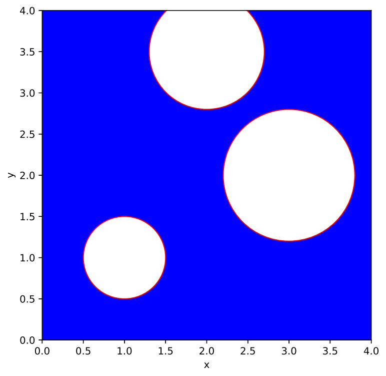
where, the coefficient $\mathbf{c} = 1.0$ and the time $T = 2$. The region Ω represents a complex geometric area, as shown in Fig. 9(a). The observational points are marked by blue dots in Fig. 9(b).

To verify the method's accuracy, we assume an exact solution of the following form:

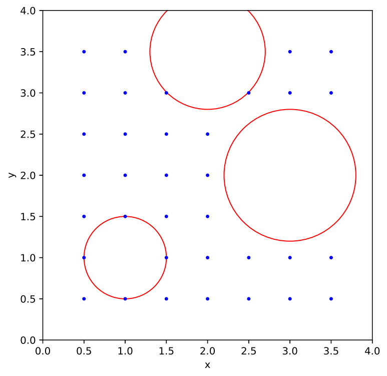
$$u_e(x,y,t) = \chi(x,y) \cdot \sin(\mu x)\sin(\nu y)[2\cos(\lambda t) + \sin(\lambda t)],$$

where

$$\mu = \frac{2\pi}{x_{\max} - x_{\min}}, \quad \nu = \frac{2\pi}{y_{\max} - y_{\min}}, \quad \lambda = \sqrt{\mu^2 + \nu^2}.$$



(a) Complex geometric domain



(b) Observation points in the complex domain

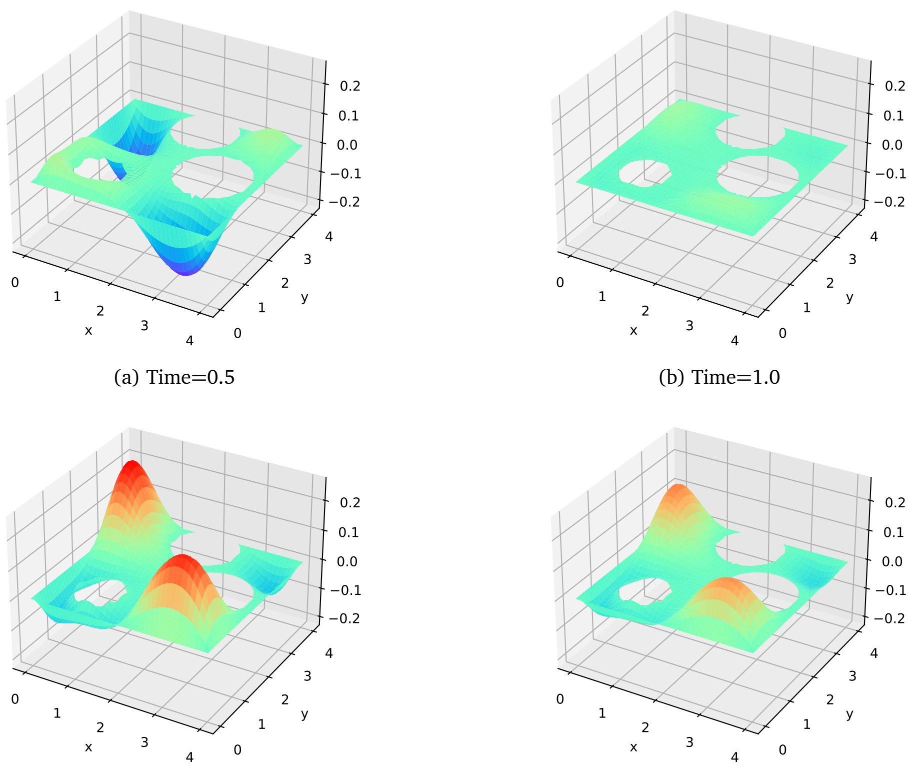
Figure 9: Complex geometric domain and configuration of observation points.

$\chi(x,y)$ is the function enforcing the sticky boundary conditions

$$\chi(x,y) = \prod_{i=1}^{n}\left[1-\exp\left\{-\alpha\cdot\frac{(x-x_i)^2+(y-y_i)^2-r_i^2}{r_i^2}\right\}\right],$$

where, x_i, y_i, and r_i represent the centers and radii of circular holes in the domain, with $\alpha = 0.1$ as the steepness coefficient. Consequently, $f(x,y,t)$ can be derived accordingly. The observation function value is computed from the exact solution as $h(x_i,y_j,t_k) = u_e(x_i,y_j,t_k)$. Based on the physical background, it is known that the boundary conditions are set to zero, and in this particular example, no initial conditions are applied.

We solve the continuous solution reconstruction problem using the IRFM with default hyperparameters set to $N_x = N_y = 4$, $N_t = 2$, and $Q_x = Q_y = Q_t = 10$. Table 5 shows the reconstruction errors for varying numbers of basis functions, while Fig. 10 displays the reconstructed solution at different time points with $N_i = 200$. This demonstrates the flexibility and robustness of the IRFM in continuous solution reconstruction for complex domains, a capability that FEM-type methods lack.



(a) Time=0.5 (b) Time=1.0

(c) Time=1.5 (d) Time=2.0

Figure 10: Reconstructed solution for the MVE in a complex domain at different times.

Table 5: Errors of IRFM based reconstructed solution for the MVE in a complex domain.

N_i	L^∞ error	L^2 error
50	7.70E-02	2.09E-02
100	9.41E-03	1.70E-03
150	2.22E-03	5.61E-04
200	1.42E-03	3.22E-04

4. Conclusion

In this paper, we present an improved RFM method to integrate sparse observational data with PDE models, aiming to reconstruct high-accuracy continuous solutions. We reformulate the data-model fusion problem as a constrained optimization problem. By using random feature approximation across multiple regions, we derive equivalent conditions for the optimal solution. Original RFM use partition of unity functions to ensure continuity at subregion boundaries. Due to that it is challenging to define PoU functions if the domain is irregularly partitioned, and the PoU function is usually piecewise defined, go against computational efficiency. Rather than using PoU function, we extend the random feature functions across the entire domain, preserving the approximation capability of the random feature functions within subregions while enhancing their global smoothness. To accurately calculate the cost associated with noisy sparse observational data, we introduce a distance-weighting strategy that eliminates interpolation errors and enhances robustness against noise.

We validate the performance and accuracy of the IRFM through numerical examples. First, we demonstrate that IRFM achieves greater precision and faster convergence than the original RFM method by solving static and dynamic PDE models. Next, we apply IRFM to reconstruct continuous solutions by integrating observational data with a reaction-convection-diffusion equation, further evidencing its effectiveness and improved accuracy over the original RFM. We also compare IRFM with PINNs, highlighting its efficiency and precision. Finally, we investigate the solution reconstruction for the membrane vibration problem in complex domains, showcasing IRFM's flexibility in tackling intricate challenges.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 12271248).

References

[1] M. Asch, M. Bocquet and M. Nodet, *Data Assimilation: Methods, Algorithms, and Applications*, SIAM (2016).

[2] J. Chen, X. Chi, W. E and Z. Yang, *Bridging traditional and machine learning-based algorithms for solving PDEs: The random feature method*, J. Mach. Learn. **1**, 268–298 (2022).

[3] J. Chen, W. E and Y. Luo, *The random feature method for time-dependent problems*, East Asian J. Appl. Math. **13**(3), 435–463 (2023).

[4] S. Cheng, C. Liu, Y. Guo and R. Arcucci, *Efficient deep data assimilation with sparse observations and time-varying sensors*, J. Comput. Phys. **496**, Paper No. 112581 (2024).

[5] R. Cintra, H. de Campos Velho and S. Cocke, *Tracking the model: Data assimilation by artificial neural network*, in: *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 403–410 (2016).

[6] A.N. Deshmukh, M.C. Deo, P.K. Bhaskaran, T.M. Balakrishnan Nair and K.G. Sandhya, *Neural-network-based data assimilation to improve numerical ocean wave forecast*, IEEE J. Ocean. Eng. **41**, 944–953 (2016).

[7] A.J. Geer et al., *All-sky satellite data assimilation at operational weather forecasting centres*, Q. J. R. Meteorol. Soc. **144**(713), 1191–1217 (2018).

[8] L. Hascoët, *Adjoints by automatic differentiation*, in: *Advanced Data Assimilation for Geosciences*, E. Blayo, M. Bocquet, E. Cosme and L.F. Cugliandolo (Eds), 349–369, Oxford University Press, (2014).

[9] Q.Z. He, D. Barajas-Solano, G. Tartakovsky and A.M. Tartakovsky, *Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport*, Adv. Water Resour. **141**, Paper No. 103610 (2020).

[10] P.L. Houtekamer and H.L. Mitchell, *Data assimilation using an ensemble Kalman filter technique*, Mon. Weather Rev. **126**, 796–811 (1998).

[11] P.L. Houtekamer and H.L. Mitchell, *A sequential ensemble Kalman filter for atmospheric data assimilation*, Mon. Weather Rev. **129**, 123–137 (2001).

[12] E. Kalnay, *Atmospheric Modeling, Data Assimilation and Predictability*, Cambridge University Press (2003).

[13] W.A. Lahoz and P. Schneider, *Data assimilation: Making sense of Earth observation*, Front. Environ. Sci. **2**, Paper No. 16 (2014).

[14] F.X. Le Dimet and I.M. Navon, *Variational and optimization methods in meteorology: A review*, Supercomput. Comput. Res. Inst. **144** (1988).

[15] Y. Liu et al., *Advancing data assimilation in operational hydrologic forecasting: Progresses, challenges, and emerging opportunities*, Hydrol. Earth Syst. Sci. **16**, 3863–3887 (2012).

[16] A. Manzoni, A. Quarteroni and S. Salsa, *Optimal Control of Partial Differential Equations*, Springer International Publishing (2021).

[17] M.J. Martin et al., *Status and future of data assimilation in operational oceanography*, J. Oper. Oceanogr. **8**(sup1), s28–s48 (2015).

[18] M. Raissi, P. Perdikaris and G.E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, J. Comput. Phys. **378**, 686–707 (2019).

[19] W. Tobler, *On the first law of geography: A reply*, Ann. Assoc. Am. Geogr. **94**, 304–310 (2004).

[20] A.T. Weaver, *3D-Var and 4D-Var approaches to ocean data assimilation*, in: *ECMWF Workshop on the Role of the Upper Ocean in Medium and Extended Range Forecasting*, Reading, United Kingdom, ECMWF, 57–66 (2003).

[21] S. Xu, Z. Sun, R. Huang, D. Guo, G. Yang and S. Ju, *A practical approach to flow field reconstruction with sparse or incomplete data through physics informed neural network*, Acta Mech. Sin. **39**, Paper No. 322302 (2023).

[22] K. Yeo, *Data-driven reconstruction of nonlinear dynamics from sparse observation*, J. Comput. Phys. **395**, 671–689 (2019).

[23] O.C. Zienkiewicz and R.L. Taylor, *The Finite Element Method*, Butterworth-Heinemann (2000).