

Interpretable Multifidelity Learning with Kolmogorov-Arnold Networks for Spatiotemporal Nuclear Reactor Flow Modeling

Meredith Eaheart^{a*}, Majdi I. Radaideh^{a,b*}

^a*Department of Nuclear Engineering and Radiological Science, University of Michigan, Ann Arbor, Michigan 48109*

^b*Department of Computer Science and Engineering, University of Michigan, Ann Arbor, Michigan 48109*

Abstract

High-fidelity computational fluid dynamics (CFD) simulations of nuclear reactor systems are computationally expensive, motivating machine learning methods that reduce cost while preserving accuracy. Multifidelity surrogate modeling addresses this by combining inexpensive low-fidelity data with limited high-fidelity data. This work investigates whether the high-fidelity data required to predict the spatiotemporal mass flow rate across all 193 fuel assemblies of a pressurized water reactor (PWR) core can be reduced, comparing black-box multifidelity feedforward neural networks (MFNNs) with interpretable multifidelity Kolmogorov-Arnold Networks (MFKANs). Cross-fidelity correlation across the core is weak, making this a challenging multifidelity problem, and neither architecture predicts the transient particularly well in absolute terms. MFKAN nonetheless performs comparably to MFNN for one-time-step-forward prediction while also providing closed-form symbolic equations unavailable with MFNN. To address the scarcity of CFD transients, with only one available per fidelity level, TabDDPM-generated synthetic data was used to augment cross-fidelity forecasting. KAN and FNN achieve comparable self-evaluation accuracy (Pearson correlation of roughly 0.73–0.81), but both perform poorly at true cross-fidelity forecasting (correlations of 0.03–0.06). Assembly-level analysis shows that each assembly’s forecast correlation tracks its real cross-fidelity correlation for both architectures ($r = 0.840$ – 0.896 for KAN, $r = 0.804$ – 0.903 for FNN), confirming that forecasting accuracy is fundamentally limited by how correlated the fidelity levels already are. Overall, KAN-based multifidelity modeling achieves accuracy consistently comparable to MFNN across all settings, while additionally providing symbolic equations that match or exceed the spline representation’s accuracy.

Keywords: Multifidelity Modeling, Kolmogorov-Arnold Networks, Computational Fluid Dynamics, Pressurized Water Reactors, Thermal-Hydraulics, Generative AI

1. Introduction

Nuclear reactors experience complex physical phenomena and require detailed simulations to capture this complexity, but that comes at a cost: high-fidelity computational fluid dynamics (CFD) simulations are accurate but expensive to run [1], while lower-fidelity simulations are cheaper but less accurate. This tradeoff motivates machine learning methods that can preserve accuracy while cutting the computational cost [2]. One technique that has been used to save computational cost is surrogate modeling and more specifically multifidelity surrogate modeling, which combines a small amount of expensive high-fidelity data with a larger amount of cheap low-fidelity data.

The problem considered in this work combines several difficulties that are rarely encountered together. The quantity of interest is the full spatiotemporal mass flow rate field at the core inlet of a pressurized water reactor (PWR), a high-dimensional, turbulence-driven transient, and the available CFD fidelity levels are only weakly correlated with one another. The CFD simulations used for multifidelity surrogate modeling in this work are also limited to a single transient at each fidelity level, restricting the amount and diversity of data available for training, and generating additional high-fidelity CFD transients is computationally expensive. At the same time, safety-relevant applications favor surrogate models whose predictions can be inspected rather than taken on trust [3]. Section 2 reviews how existing work addresses each of these difficulties and identifies the gaps that remain.

*Corresponding Author: M. Eaheart (eaheart@umich.edu), M. I. Radaideh (radaideh@umich.edu)

This work develops and systematically evaluates multifidelity feedforward neural networks (MFNNs) and multifidelity Kolmogorov-Arnold Networks (MFKANs) for predicting the full spatiotemporal mass flow rate field in a large-scale PWR core. Three CFD mesh-based fidelity levels derived from the same underlying reactor model [2] are used to investigate multifidelity learning under weak cross-fidelity correlation and severely limited transient data. Beyond benchmarking established multifidelity neural-network approaches, this work introduces new KAN-based architectures designed to combine multifidelity prediction with symbolic interpretability. The main contributions are:

- Full-core spatiotemporal multifidelity prediction: MFNN and MFKAN methods are applied, for the first time, to full-core spatiotemporal mass flow rate prediction across all 193 PWR fuel assemblies, extending multifidelity surrogate modeling from scalar or low-dimensional quantities to a high-dimensional, time-dependent reactor field.
- New fidelity-conditioned KAN architecture: A new multifidelity architecture, *MFKAN-Flag*, is introduced in which the fidelity level is explicitly encoded as an input to a unified KAN. MFKAN-Flag is benchmarked against MFKAN-Delta and conventional MFNN approaches to determine whether KAN-based multifidelity models can achieve comparable predictive accuracy while additionally yielding interpretable symbolic representations unavailable from standard black-box neural networks.
- Extension to three fidelity levels: *MFKAN-Flag3f* extends the proposed fidelity-conditioned architecture from two to three CFD fidelity levels, demonstrating how a single KAN framework can simultaneously learn information across multiple mesh resolutions rather than being restricted to a conventional low-/high-fidelity pair.
- Generative augmentation for data-scarce multifidelity CFD: TabDDPM-based synthetic data generation is integrated with the multifidelity framework to address the extreme scarcity of CFD transient data, with only one transient available at each fidelity level. The augmented datasets are used for cross-fidelity transient forecasting, providing a pathway to exploit generative models when additional high-fidelity CFD simulations are prohibitively expensive.

These contributions establish and evaluate an interpretable KAN-based framework for high-dimensional multifidelity reactor prediction, while addressing two central challenges of practical CFD surrogate modeling: weak correlations between fidelity levels and severe scarcity of high-fidelity transient data. The remainder of this paper is organized as follows. Section 2 reviews related work and identifies the gaps addressed here. Section 3 describes the dataset and the machine learning and multifidelity methods considered in this work. Section 4 presents the results of the corresponding experiments, while Section 5 discusses the key findings, study limitations, and directions for future work, and Section 6 summarizes the main conclusions.

2. Related Work

Surrogate modeling is a widely used machine learning technique for reducing the computational cost of expensive physics-based simulations [4, 5, 6], and it has been applied across a wide range of problems in nuclear engineering, including model calibration [7], design under uncertainty constraints [8], and design and operation optimization [9]. Optimization studies in particular have used evolutionary, swarm, and reinforcement-learning-guided algorithms for tasks such as nuclear fuel design [10], microreactor reactivity control system operation [11], and general neuroevolution-based engineering design [12]. Because these algorithms typically require thousands

of model evaluations, fast surrogates are often what makes them computationally feasible. Multifidelity surrogate modeling methods extend this idea further by combining a small amount of expensive high-fidelity data with a larger amount of cheap low fidelity data, an approach with a long history in engineering more broadly [13, 14]. These multifidelity surrogate models are most commonly built as either multifidelity neural networks (MFNN) [15, 16, 17] or multifidelity Gaussian processes [18, 19]. Our own prior work directly compared these two architectures for a nuclear thermal-hydraulics application [20].

Multifidelity surrogate modeling has also been applied extensively outside nuclear engineering, particularly in fluid mechanics, where it has supported tasks such as aerodynamic shape optimization [21], turbomachinery flow prediction [22], and CFD-based thermal design optimization [23]. Within nuclear engineering specifically, multifidelity methods have been used for thermal hydraulic analysis [24, 20] and reactor core simulations [25].

Multifidelity modeling is not beneficial in every setting, however. Systematic studies have shown that its advantage over single-fidelity surrogates depends strongly on how well the fidelity levels are correlated, and can disappear when the low-fidelity model is a poor predictor of the high-fidelity one [26, 27]. Nuclear multifidelity applications to date have mostly targeted scalar or low-dimensional quantities of interest, and little is known about how these methods behave when cross-fidelity correlation is weak.

The coolant flow distribution at the inlet of a PWR core is shaped by mixing in the downcomer and lower plenum, and its prediction has long relied on CFD validated against dedicated experiments [28]. Resolving the turbulence-driven fluctuations of this flow across a full core at fine temporal resolution is particularly expensive [2]. Machine learning has become a common route to reducing such costs in fluid mechanics [29], typically by compressing the flow field into a small number of modes and learning how those modes evolve in time. These reduced-order models follow two broad designs: static feedforward networks that map the current modal state to the next one [30], and recurrent architectures, such as long short-term memory (LSTM) networks, that carry information across longer histories [31, 32, 33].

In nuclear engineering, deep neural networks and reduced-order models have been used to predict high-dimensional, core-wide quantities such as power distributions, neutron flux, and three-dimensional temperature fields [34, 35, 36]. For the same PWR lower-plenum model used in this work, machine learning has also been applied to flow-field reconstruction and one-step-ahead forecasting [2]. These high-dimensional studies, however, rely on a single source of training data. To our knowledge, multifidelity learning has not yet been applied to full-core spatiotemporal fields, and the trade-off between static and recurrent architectures has not been considered in settings where the resulting model must also be interpretable.

Despite these successes, the multifidelity methods described above are typically built on classical feedforward neural networks, which offer little insight into why a given prediction was made [37], or Gaussian processes, which provide useful uncertainty quantification but not the same closed-form interpretability offered by symbolic regression. For safety-relevant applications, models that are interpretable by construction are generally preferable to post hoc explanations of black-box models [3]. Symbolic approaches pursue this goal by expressing the learned model as an explicit equation, ranging from polynomial networks built with the group method of data handling [38] to sparse identification of governing equations [39] and neural-network-guided symbolic regression [40].

Kolmogorov-Arnold Networks (KAN) [41] offer an interpretable alternative to these neural-network architectures, since a trained KAN can be converted into a closed-form symbolic equation that can be more easily understood by a human. KANs have since been applied to nuclear engineering problems, including interpretable regression for reactor operation and safety applications [37, 42], as well as to fluid mechanics problems such as two-phase flow modeling [43], turbulence anisotropy mapping [44], and uncertainty aware flow field reconstruction [45]. These interpretable surrogates, however, have been trained almost exclusively on single-fidelity data.

KAN has recently been extended into the multifidelity setting [46]. Multifidelity KAN (MFKAN) combines these two ideas: it was first introduced as a low-fidelity KAN paired with a correction network that learns the residual toward high-fidelity data [46]. Because a trained KAN can be converted to a symbolic equation, MFKAN also opens the door to interpretable, closed-form multifidelity models; concurrent work has explored applying symbolic regression directly within a KAN-based multifidelity data-fusion framework [47].

Whether KANs improve on conventional multilayer perceptrons is itself still debated: controlled comparisons on scientific problems report that some KAN variants match multilayer perceptrons in accuracy, while others are less robust to training conditions [48]. For a multifidelity KAN, the practical question is therefore whether it can match a conventional MFNN while additionally providing a symbolic model. Existing MFKAN formulations follow a residual-correction structure with two fidelity levels. To our knowledge, a single KAN conditioned directly on the fidelity level, its extension to more than two fidelity levels, and a direct comparison of MFKAN against MFNN on a large-scale engineering field have not been reported.

When generating additional high-fidelity CFD transients is computationally expensive, generative AI offers a way to augment the available data with representative synthetic samples. Generative adversarial networks have been adapted to tabular data [49] and to time series while preserving their temporal dynamics [50], and denoising diffusion probabilistic models [51] underlie TabDDPM [52], a diffusion-based generative model designed for synthetic tabular data, which is used in this work. A broader overview of time-series augmentation strategies is given in [53]. Similar generative data-augmentation approaches have also been explored in nuclear engineering and energy applications [54, 55], supporting their potential to expand limited simulation datasets without requiring additional costly CFD calculations. Complementary strategies for limited data include uncertainty-aware variational neural networks [56] and active learning, which reduces the number of expensive simulations by selecting only the most informative ones to run [57]. These approaches have been developed in single-fidelity settings; their use to supply training data for multifidelity learning, when only one transient exists at each fidelity level, has not been explored.

In summary, three gaps motivate this work. First, multifidelity learning has not been applied to full-core spatiotemporal reactor fields, particularly under weak cross-fidelity correlation. Second, interpretable multifidelity models remain limited to residual-type, two-fidelity KAN formulations. Third, generative augmentation has not been combined with multifidelity learning to overcome the scarcity of CFD transients. Section 3 describes the dataset and the methods used to address these gaps.

3. Methods

3.1. CFD Simulation and Data Generation

The CFD data set used in this work comes from a prior study done by our team [2], which built a full-scale four-loop PWR model. Swirl from the centrifugal reactor coolant pumps combined with chaotic turbulent flow at the core inlet produces localized mass flow rate perturbations across the fuel assemblies that matter operationally because they can drive power tilts, giving this prediction problem safety significance. Resolving this behavior with a high-fidelity CFD transient is expensive, motivating the multifidelity approach used here.

3.1.1. Full-Scale PWR Lower-Plenum CFD Model

The model, created in STAR-CCM+, covers the four cold-leg inlets down through the downcomer and lower plenum and ends at the core inlet, with geometry built from the publicly available BEAVRS benchmark [58]. The 193 fuel assemblies are modeled as porous media using a Forchheimer resistance formulation, and swirl is imposed at the four cold-leg inlets through a user-defined boundary condition to capture the non-uniform inlet conditions typical of centrifugal reactor coolant pumps.

Turbulence is modeled with the STRUCT- ε closure. The complete details of the CFD model, the construction of the mesh and the formulation of turbulence are given in [2]; only the details needed to understand the machine learning tasks covered in this research are summarized here.

3.1.2. Transient Dataset and Quantity of Interest

Each transient runs for 15 s of physical time at a fixed 0.0005 s timestep. The first 10 s are discarded to let the flow field develop, and data are recorded over the remaining 10 to 15 s window, giving 10,000 timesteps per transient. The recorded quantity is the area-averaged mass flow rate at each of the 193 fuel assemblies, sampled at nine axial layers of the CFD mesh spanning the height of the core. This paper uses only the lowest of these nine axial layers, where the inlet flow retains the strongest swirl-driven heterogeneity and is therefore the most chaotic and difficult layer to predict. Figure 1 shows how the mass flow rate changes axially as the flow travels upward through the core: it varies substantially at the bottom layer, then homogenizes quickly higher in the core. *We hypothesize that flow prediction is inherently more challenging in the base layer, consistent with the findings of [2], where the base-layer models exhibited substantially higher prediction errors than the upper layers for both flow-field reconstruction and one-step-ahead forecasting.*

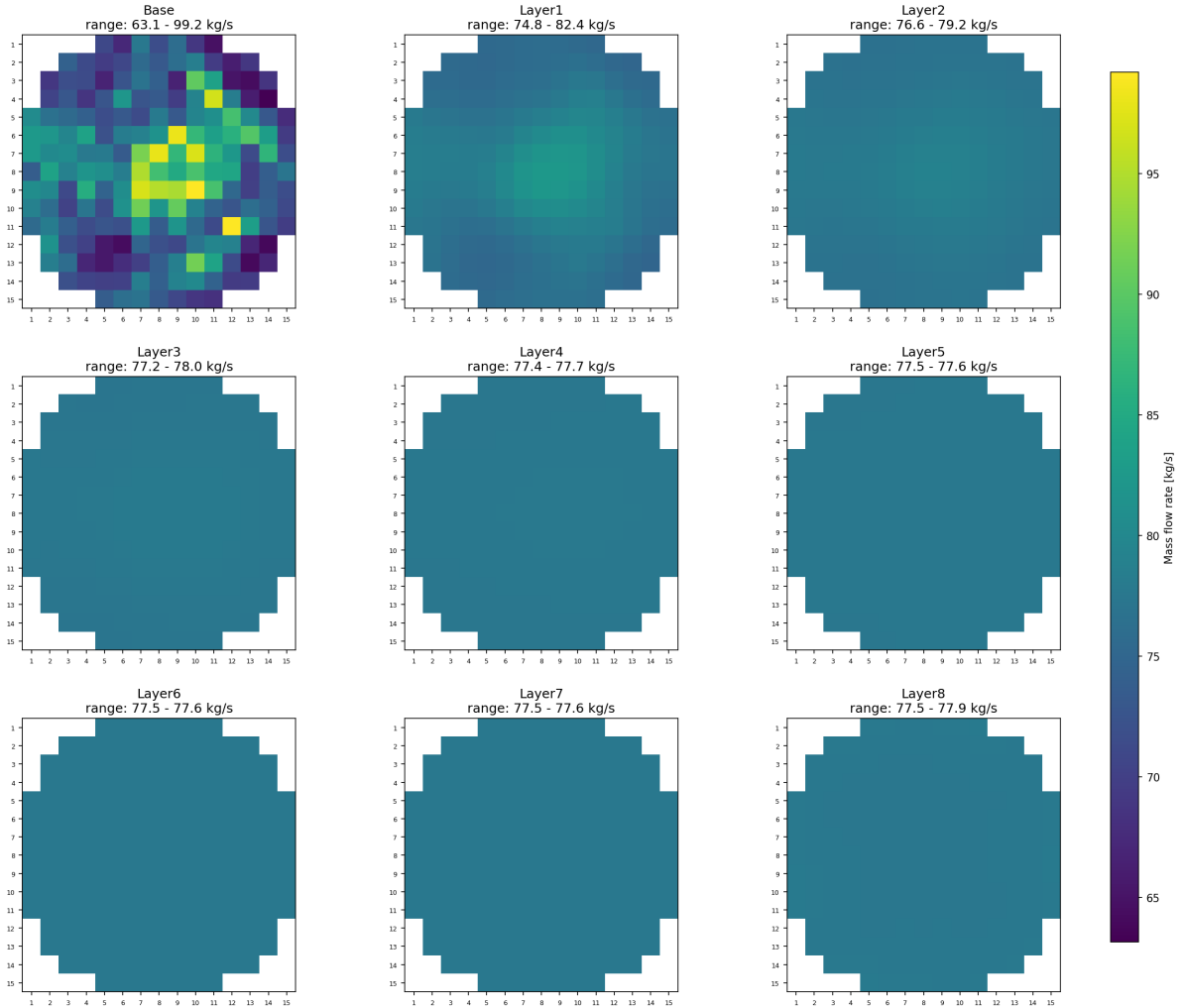


Figure 1: Mass flow rate at the beginning of the usable transient across all nine axial layers, shown using a common color scale for direct comparison. *Note the substantial flow heterogeneity in the base layer, which is the only layer examined further in this work.*

3.1.3. Fidelity Level Definition

Three meshes at different resolutions, low, medium, and high-fidelity, are built by scaling the fuel-assembly pitch element size to base sizes of 2.20 cm, 1.83 cm, and 1.57 cm, respectively. Everything else about the model, the geometry, boundary conditions, and solver settings, stays the same across all three settings. *These three meshes, denoted FApitch10, FApitch12, and FApitch14 in the source CFD study [2], are treated as the LF (low-fidelity), MF (medium-fidelity), and HF (high-fidelity) datasets used throughout this work.* Running the high-fidelity mesh on 200 cores took roughly **20 days**, and the low-fidelity mesh was done in about **6 days** on the same hardware, a gap that makes clear just how costly the high-fidelity transient is by comparison. Only one transient is run at each fidelity level due to substantial computational constraints.

3.2. One-Time-Step-Forward Prediction Framing

This transient is chaotic and highly oscillatory due to turbulence, with limited temporal dependence from one timestep to the next, making long-term forecasting difficult. Similar to the temporal methods used in [2], this work formulates prediction by pairing the state at time t with the state at time $t + 1$. However, static methods are used instead of sequence-based architectures, such that each input-output pair is treated as an independent training example rather than as part of a longer temporal sequence. We hypothesize that recurrent methods, such as long short-term memory (LSTM) variants, would provide little additional benefit for this problem because of the weak dependence on previous timesteps, while introducing greater network complexity and training cost compared with static methods.

Both the input and the target are 193-dimensional vectors, one value per fuel assembly at the bottom axial layer described in Section 3.1.2. At test time, the input at each step is the true state from the CFD transient rather than a state predicted by the model at a previous step, so error does not compound the way it would in a genuine multi-step rollout.

Training and test sets are split chronologically rather than shuffled, so no information from later in the transient leaks into training. High-fidelity data, the actual prediction target, uses the first 20% of its transient for training and the remaining 80% for testing. Low-fidelity data (and medium-fidelity data, for the three-fidelity methods) is used in full for training, since it only serves as auxiliary input rather than the quantity being evaluated.

This one-time-step-forward framing is used unchanged for the experiments described in Section 3.6 and Section 3.7. It does not apply to the TabDDPM-augmented forecasting direction described in Section 3.9, which is a genuine multi-step forecasting task.

3.3. Principal Component Analysis (PCA)

Principal component analysis (PCA) is a dimensionality reduction technique. The methods in this work operate on 193-dimensional assembly-level inputs and outputs, which is too wide to train efficiently and, for the KAN-based methods, interpretably: both KAN parameter count and symbolic regression complexity grow quickly with input/output size. PCA is therefore used as a preprocessing step, fit on the high-fidelity data and applied to compress both the input and the target before training, and as a postprocessing step, inverse transforming predictions back into physical, per-assembly units after inference. Both MFNN and MFKAN use PCA with the same convention. The number of components used varies by experiment and is reported alongside the corresponding results (Section 4). Because training is performed in PCA-compressed space, the symbolic-regression equations extracted from KAN (Sections 3.6.2 and 3.7.2) are expressed in terms of PCA components rather than the raw physical variables, allowing the resulting equations to remain compact. However, the analyst can still provide inputs and obtain predictions in the original physical space by applying the pre-computed PCA transformation and inverse transformation parameters.

3.4. Cross-Fidelity Correlation Analysis

Cross-fidelity correlation refers to how strongly the mass flow rate at one mesh resolution tracks mass flow rate at another resolution, and it matters here because conventional multifidelity methods generally perform well when fidelities are strongly correlated [13]. Visual inspection of the LF, MF, and HF transients suggested weak correlation between fidelities, so Pearson correlation [59], was used to check that impression: correlation is computed per fuel assembly i for each fidelity pair (LF-HF, MF-HF, LF-MF), giving per-assembly correlations r_i for $i = 1, \dots, 193$. Whole-core aggregate statistics are then reported as the minimum, mean, and maximum of these 193 per-assembly correlation values, i.e. $\min_i r_i$, $\frac{1}{193} \sum_{i=1}^{193} r_i$, and $\max_i r_i$.

3.4.1. Correlation-Based Assembly Subset

The correlated-assembly subset selects the 50 assemblies with the highest positive LF-HF Pearson correlation (Section 3.4), and restricts both the input and output of the MFNN/MFKAN methods (Section 3.6) to this subset, PCA-compressed to 10 components, rather than the full 193-assembly core. This produces a smaller, more self-correlated prediction task that is not directly comparable to the full-core baseline (Section 4.3.2), though it directly tests whether stronger cross-fidelity correlation improves multifidelity prediction accuracy, complementing rather than replacing the full-core baseline. This subset also underlies the separate assembly-level analysis in Section 4.4.5, which checks whether TabDDPM’s best-forecasting assemblies coincide with the most-correlated ones.

3.5. Spectral Clustering

Spectral clustering is a graph-based clustering approach that identifies groups from pairwise similarities rather than directly partitioning points in their original feature space [60]. In the normalized spectral clustering algorithm of Ng, Jordan, and Weiss [61], the data are represented by a weighted similarity graph, from which the symmetric normalized graph Laplacian, L_{sym} , is constructed. The first k eigenvectors corresponding to its smallest eigenvalues form a spectral embedding, whose rows are normalized to unit length and subsequently partitioned using k -means clustering. This transformation provides a representation in which the underlying cluster structure can become more readily separable, including for data with complex or non-convex structure.

Clusters are formed from HF transients using the spectral clustering approach described above, producing four clusters. Each network’s input still sees the full 193-assembly LF (and MF, for three-fidelity methods) core, while its output is restricted to its own cluster’s assemblies, so four separate networks are trained per method. Results are reported in Section 4.3.3.

3.6. Machine Learning Methods

3.6.1. Feedforward Neural Networks (FNN)

A feedforward neural network (FNN) [62] approximates a function by composing layers of nodes, where each node applies a fixed nonlinear activation function to a weighted sum of its inputs from the previous layer. The weights and biases are learned via backpropagation to minimize a loss function on the training data. FNNs serve as the standard general-purpose function approximator used here and form the building block for the multifidelity FNN baseline described in Section 3.7.1.

3.6.2. Kolmogorov-Arnold Networks (KAN)

A Kolmogorov-Arnold Network (KAN) [41] differs from a standard feedforward neural network (FNN) in how nonlinear transformations are represented and learned. In an FNN, connections are parameterized by trainable scalar weights, while fixed nonlinear activation functions are applied at the nodes. In contrast, KAN replaces these scalar weights with trainable univariate functions, typically parameterized using splines, while each node sums the

outputs of its incoming functions. Thus, nonlinear transformations are learned directly on the network connections rather than imposed through fixed node activation functions. Because these learned functions are explicit and univariate, they can be inspected, simplified, and approximated by analytical functions after training, allowing the trained network to be expressed as a closed-form symbolic model through symbolic regression. This enhanced interpretability motivates the use of KAN throughout this work and forms the basis for the multifidelity KAN formulations introduced in Section 3.7.2.

After training, the learned KAN spline functions can be approximated by closed-form symbolic expressions using `pykan's auto_symbolic()` procedure. This procedure fits a library of candidate elementary functions to each trained spline and replaces the spline with the best-matching symbolic form. Symbolic regression is performed on the same PCA-compressed inputs and outputs used during training (Section 3.3), rather than on the raw physical variables. Throughout this work, the resulting symbolic model is evaluated on the same test set as the original spline-based model, allowing their predictive accuracies to be compared directly.

3.7. Multifidelity Methods

3.7.1. Multifidelity FNN (MFNN)

Three multifidelity formulations were investigated, following the same Delta and Flag implementations and structures used in our prior work [20], adapted here as neural-network baselines for the MFKAN comparison in Section 3.7.2. These baselines establish whether conventional multifidelity learning can exploit whatever cross-fidelity structure is present in this dataset (Section 3.4).

MFNN: Delta Method. The Delta method [15] models the high-fidelity response as a correction to a learned low-fidelity prediction. First, an FNN $f_L(x)$ is trained on the low-fidelity dataset. The trained model is evaluated at the high-fidelity input locations, and the discrepancy between the high-fidelity observations and these predictions is computed as

$$r(x) = y_H(x) - f_L(x) \quad (1)$$

where x denotes the high-fidelity input locations. A second FNN $f_\Delta(x, f_L(x))$ is then trained to model this residual at those same high-fidelity locations, and the final prediction is

$$\hat{y}_H(x) = f_L(x) + f_\Delta(x, f_L(x)) \quad (2)$$

The two networks are trained sequentially and do not require the low- and high-fidelity datasets to share identical sampling locations.

MFNN: Flag Method. In the fidelity-flag approach [16], a single FNN is trained to learn multiple fidelity mappings simultaneously. The fidelity level is incorporated as an additional input feature by appending a fidelity indicator f to the input vector, with $f = 0$ denoting low-fidelity and $f = 1$ denoting high-fidelity data. High-fidelity predictions are obtained by evaluating the learned mapping $F(x, f)$ at $f = 1$,

$$\hat{y}_H(x) = F(x, 1) \quad (3)$$

Correlations between fidelity levels are learned implicitly through the network's shared parameters.

MFNN: Three-Fidelity Flag Method. The Flag method extends naturally to three fidelity levels by generalizing the fidelity indicator to $f \in \{0, 1, 2\}$ for low-, medium-, and high-fidelity data, respectively. High-fidelity predictions are

obtained by evaluating $F(x, f)$ at $f = 2$,

$$\hat{y}_H(x) = F(x, 2) \quad (4)$$

3.7.2. Multifidelity KAN (MFKAN)

Three multifidelity KAN formulations were investigated, following the same Delta and Flag structures described for MFNN above (Section 3.7.1), with each FNN replaced by a KAN. After an initial fit, each KAN is pruned, then retrained, following pykan’s standard training convention.

MFKAN: Delta Method. The Delta method follows the same construction as MFNN-Delta above (Section 3.7.1, Eqs. (1)–(2)), with f_L and f_Δ implemented as KANs instead of FNNs.

Symbolic regression (Section 3.6.2) is applied independently to the trained f_L and f_Δ KANs, converting each network’s learned splines into its own closed-form symbolic expression, f_L^{sym} and f_Δ^{sym} respectively. These two symbolic models are then chained the same way as their numeric forward pass to produce the final symbolic prediction:

$$\hat{y}_H^{\text{sym}}(x) = f_L^{\text{sym}}(x) + f_\Delta^{\text{sym}}(x, f_L^{\text{sym}}(x)). \quad (5)$$

That is, f_L ’s symbolic expression is evaluated first, and its output is fed into f_Δ ’s symbolic expression as part of its input.

MFKAN: Flag Method (Proposed). The Flag method follows the same construction as MFNN-Flag above (Section 3.7.1), with F implemented as a single KAN instead of an FNN, which we propose here as a new MFKAN variant. Symbolic regression (Section 3.6.2) is applied once to the single trained KAN, producing one symbolic model with the fidelity flag as an input variable alongside the PCA-compressed inputs (Section 3.3). Only one lossy spline-to-symbolic conversion is needed here, unlike MFKAN-Delta’s two symbolic conversions.

MFKAN: Three-Fidelity Flag Method (Proposed). The three-fidelity Flag method, which we call *MFKAN-Flag3f*, extends the two-fidelity MFKAN-Flag method proposed above to three fidelity levels, following the same construction as MFNN’s three-fidelity Flag method above (Section 3.7.1), with F implemented as a single KAN instead of an FNN. As with MFKAN-Flag, symbolic regression is applied once to the single trained KAN, with the fidelity flag as an input variable alongside the PCA-compressed inputs.

3.7.3. Implementation Validation on Analytical Benchmarks

Before being applied to the reactor CFD dataset, the MFKAN-Delta and MFKAN-Flag implementations were validated on the same two-fidelity analytical benchmark functions used in [20]: the bi-fidelity Forrester function [63] and the bi-fidelity Branin function, both drawn from the MF2 package [64]. MFKAN-Flag3f was similarly validated on three-fidelity Forrester (1D), Rastrigin (2D), and Rosenbrock (2D) functions [65]. These benchmarks have known analytical solutions, so they confirm each implementation reproduces sensible multifidelity behavior before being applied to the real-world engineering dataset, which has no such ground truth to check against directly.

Bi-fidelity Forrester Function (1D). The bi-fidelity Forrester function is derived from the classical one-dimensional Forrester test function [63] as shown below:

$$f_h(x) = (6x - 2)^2 \sin(12x - 4) \quad (6)$$

$$f_l(x) = Af_h(x) + B(x - 0.5) + C \quad (7)$$

where $x \in [0, 1]$. The recommended parameters are:

$$A = 0.5, \quad B = 10, \quad C = -5$$

Bi-fidelity Branin Function (2D). The bi-fidelity Branin function is defined by [64] as

$$f_b(x_1, x_2) = \left(x_2 - 5.1 \frac{x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6 \right)^2 + 10 \cos(x_1) \left(1 - \frac{1}{8\pi} \right) + 10, \quad (8)$$

$$f_h(x_1, x_2) = f_b(x_1, x_2) - 22.5x_2, \quad (9)$$

$$f_l(x_1, x_2) = f_b(0.7x_1, 0.7x_2) - 15.75x_2 + 20(0.9 + x_1)^2 - 50, \quad (10)$$

where $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$.

Multi-fidelity Forrester (Adapted to Tri-Fidelity). The multifidelity Forrester function is defined as shown below [65, 63]:

$$f_1(x) = (6x - 2)^2 \sin(12x - 4), \quad (11)$$

$$f_2(x) = (5.5x - 2.5)^2 \sin(12x - 4), \quad (12)$$

$$f_3(x) = 0.75(6x - 2)^2 \sin(12x - 4) + 5(x - 0.5) - 2, \quad (13)$$

$$f_4(x) = 0.5(6x - 2)^2 \sin(12x - 4) + 10(x - 0.5) - 5, \quad (14)$$

where $x \in [0, 1]$. This formulation defines four fidelity levels, ordered from highest (f_1) to lowest (f_4), but only three are needed here. f_1 is therefore omitted, and the remaining three are reassigned as high-fidelity (f_2), medium-fidelity (f_3), and low-fidelity (f_4).

Tri-fidelity Rosenbrock. The tri-fidelity Rosenbrock function is defined as shown below [65, 66, 67]:

$$f_1(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right] \quad (15)$$

$$f_2(\mathbf{x}) = \sum_{i=1}^{D-1} \left[50(x_{i+1} - x_i^2)^2 + (-2 - x_i)^2 \right] - \sum_{i=1}^D 0.5x_i \quad (16)$$

$$f_3(\mathbf{x}) = \frac{f_1(\mathbf{x}) - 4 - \sum_{i=1}^D 0.5x_i}{10 + \sum_{i=1}^D 0.25x_i} \quad (17)$$

$$\mathbf{x} \in [-2, 2]^D$$

Similarly to the representation in the Forrester function, f_3 represents the lowest fidelity, f_2 is the medium-fidelity, and f_1 is the high-fidelity solution. One interesting feature of these equations is that the number of input dimensions can be changed by refining the D parameter. In this work, $D = 2$ was used to validate the performance of the three-fidelity methods.

Tri-fidelity Rastrigin. The tri-fidelity shifted-rotated Rastrigin function [67, 68] is defined as

$$f_i(\mathbf{z}; \phi_i) = f_1(\mathbf{z}) + e_r(\mathbf{z}, \phi_i), \quad i = 1, 2, 3, \quad (18)$$

where the base Rastrigin function is

$$f_1(\mathbf{z}) = \sum_{j=1}^D (z_j^2 + 1 - \cos(10\pi z_j)), \quad (19)$$

and

$$\mathbf{z} = R_D(\theta)(\mathbf{x} - \mathbf{x}^*). \quad (20)$$

For the two-dimensional case used in this work,

$$R_2(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}. \quad (21)$$

The fidelity-dependent error term is

$$e_r(\mathbf{z}, \phi) = \sum_{j=1}^D a(\phi) \cos^2(w(\phi)z_j + b(\phi) + \pi), \quad (22)$$

with

$$a(\phi) = \Theta(\phi), \quad (23)$$

$$w(\phi) = 10\pi\Theta(\phi), \quad (24)$$

$$b(\phi) = 0.5\pi\Theta(\phi), \quad (25)$$

$$\Theta(\phi) = 1 - 0.0001\phi. \quad (26)$$

The input domain is

$$-0.1 \leq x_i \leq 0.2, \quad i = 1, \dots, D,$$

the rotation angle is $\theta = 0.2$, and the optimum occurs at

$$\mathbf{x}^* = (0.1, \dots, 0.1)^\top,$$

for which the highest-fidelity objective value is zero. The fidelity parameters are

$$\phi_1 = 10000 \quad (\text{high-fidelity}), \quad \phi_2 = 5000 \quad (\text{medium-fidelity}), \quad \phi_3 = 2500 \quad (\text{low-fidelity}).$$

3.8. Generative Models for Data Augmentation

Because only one CFD transient is currently available at each fidelity level, the limited diversity of the training data presents a major challenge for transient prediction. Generating hundreds or thousands of additional CFD transients to expand the training space is computationally prohibitive, motivating synthetic-data augmentation as a practical alternative. *The last leg of this work therefore investigates whether generative models can extract the statistical structure of the available CFD data and generate additional representative samples that improve transient prediction when real training data are severely limited.*

3.8.1. TabDDPM Method

TabDDPM [52], a diffusion-based generative model for tabular data, is used to generate additional synthetic transients at each fidelity level, addressing the scarcity of real CFD data (only one transient available per fidelity, Section 3.1.2). TabDDPM is trained once per fidelity level, then sampled repeatedly without retraining. Each newly generated sample is checked against every assembly’s real trajectory and added to whichever assembly’s bucket it correlated most strongly with, continuing until every assembly has accumulated 1,000 synthetic realizations. The computational cost of this matching process varied substantially by assembly; see Section 4.4.1 for details.

3.8.2. Synthetic Data Validation

Synthetic data quality is validated using the same Pearson correlation metric introduced in Section 3.4, computed here between each assembly’s synthetic realizations and that assembly’s own real trajectory at the same fidelity, rather than across fidelities.

3.9. TabDDPM-Augmented Forecasting Experimental Design

Synthetic realizations from Section 3.8.1 are used to train single-fidelity KAN and FNN forecasting models (Section 3.6). Two types of experiments are run, both tested with $N = 1$ and $N = 10$: the first N timesteps of the real transient being predicted are given to the model as known values, and the model must forecast the remaining timesteps on its own. In self-evaluation, a model is trained on one fidelity’s synthetic data and evaluated on that same fidelity’s real transient. In cross-fidelity forecasting, a model is trained on one or more fidelities’ synthetic data and evaluated on a different fidelity’s real transient, across three directions: LF→HF, MF→HF, and LF+MF→HF. Symbolic regression (Section 3.6.2) is applied separately to these forecasting-specific KAN models, which are newly trained for this cross-fidelity setting rather than reused from the one-time-step-forward models in Section 3.6. The resulting symbolic equations are compared against their own spline representations within this same forecasting setting, extending the spline-vs-symbolic comparison from Section 4.3.5 to this task.

3.10. Hyperparameter Tuning

Hyperparameters were tuned separately for MFKAN, MFNN (Sections 3.7.1–3.7.2), and the dedicated KAN and FNN forecasting models (Section 3.9), using Bayesian optimization in Optuna [69]. MFKAN and MFNN were tuned on the one-time-step-forward baseline case (Section 4.3.2): the 20% high-fidelity training / 80% testing split used throughout Section 4.3. The KAN and FNN forecasting models were tuned on the self-evaluation setting (e.g. LF→LF and MF→MF, $N = 1$; Section 4.4.3), then applied across the full set of forecasting directions. Table 1 summarizes the full search space and which values were held fixed.

All four searches ran for up to 100 trials, with progress checkpointed so a search could be resumed without restarting. The two forecasting searches train on the much larger synthetic TabDDPM dataset, making each trial more expensive, so they were stopped early once improvement became marginal, and the FNN forecasting search also used a lower range of training epochs than MFNN’s (Table 1). The combined-fidelity forecasting models (LF+MF→HF) were not tuned separately; they reuse the dedicated low-fidelity model’s winning hyperparameters. Final tuned values are reported alongside their respective results, in Sections 4.3.1 and 4.4.2.

Table 1: Hyperparameter search space for all four tuned model families. MFKAN/MFNN (baseline) ran up to 100 Optuna TPE trials per method to completion. Model group KAN covers MFKAN and KAN forecasting; FNN covers MFNN and FNN forecasting.

Model	Hyperparameter	Search Space
KAN	Depth (hidden layers)	$\{1, 2\}$
	Hidden width (MFKAN)	8 (fixed, not tuned)
	Hidden width (KAN forecast.)	32 (fixed, not tuned)
	Grid size	$\{3, 4, 5, 6, 7, 8, 9, 10\}$
	Spline order k	$\{2, 3, 4, 5, 6, 7, 8\}$
	Steps	$\{25, 50, 75, 100, 125, 150, 200, 250\}$
	λ	$[0, 0.001]$
	$\lambda_{entropy}$	$[0, 10]$
	Learning rate	$[10^{-4}, 10^{-1}]$ (log)
	Reg. metric	edge_forward_spline_n, edge_forward_sum, edge_forward_spline_u, edge_backward, node_backward
FNN	Hidden layers (count)	$\{2, 3, 4\}$
	Nodes per layer	$\{8, 16, 32, 64, 128, 256\}$
	Learning rate	$[10^{-5}, 10^{-1}]$ (log)
	Epochs (MFNN)	$\{250, 500, 1000, 1500, 2000, 3000, 4000, 5000\}$
	Epochs (FNN forecast.)	$\{100, 200, 300, 400, 500, 650, 800, 1000\}$
	Weight decay	$[10^{-6}, 10^{-2}]$ (log)

4. Results

Results are organized into four subsections. Section 4.1 covers the analytical-benchmark validation of the MFKAN implementations. Section 4.2 covers the cross-fidelity correlation analysis, which is one of the primary drivers of multifidelity method performance and motivates the two experimental designs that follow. Section 4.3 covers the one-time-step-forward prediction method, including the tuned hyperparameters used (Section 4.3.1) and results on the baseline (Section 4.3.2), clustered (Section 4.3.3), and most-correlated-subset (Section 4.3.4) assembly groupings, with the symbolic regression results covered in Section 4.3.5. Section 4.4 covers the TabDDPM data augmentation study, including synthetic data generation and validation (Section 4.4.1), the tuned hyperparameters used (Section 4.4.2), augmented forecasting results (Section 4.4.3), symbolic regression (Section 4.4.4), and assembly-level analysis (Section 4.4.5).

4.1. Benchmark Results

Section 3.7.3 describes the analytical-benchmark validation performed on each MFKAN implementation before it was applied to the reactor CFD dataset. Figure 2 shows parity plots and residual histograms for MFKAN-Flag and MFKAN-Delta on the bi-fidelity Branin function, and Figure 3 shows the same for MFKAN-Flag3f on the three-fidelity Forrester (1D), Rastrigin (2D), and Rosenbrock (2D) functions. The five method/benchmark combinations shown span $R^2 = 0.908$ to 0.9997 , and the bi-fidelity Forrester case (not plotted for brevity, since both methods fit it near-perfectly) reaches $R^2 = 0.9997$ for MFKAN-Flag and $R^2 = 0.9999$ for MFKAN-Delta. MFKAN-Delta on Branin is the weakest of the five ($R^2 = 0.908$), visibly noisier than the other combinations, possibly reflecting error accumulation across its two sequentially trained networks, whereas MFKAN-Flag learns both fidelities within a single network. No extensive hyperparameter search was performed in this section, as the objective was to demonstrate proof of concept and verify the implementation rather than optimize predictive

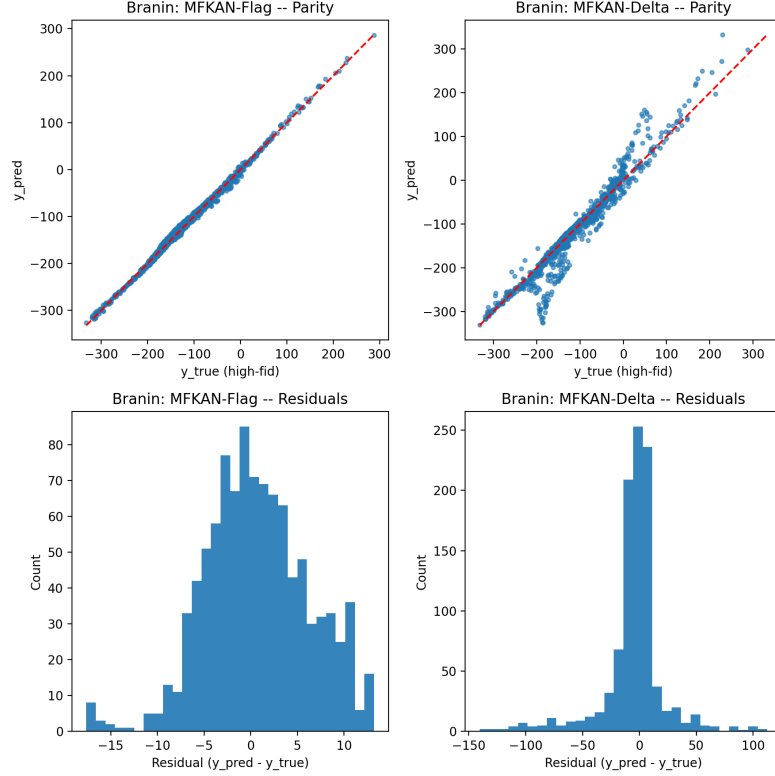


Figure 2: MFKAN-Flag and MFKAN-Delta parity plots (top) and residual histograms (bottom) on the bi-fidelity Branin function.

performance. Together, these results confirm that each implementation reproduces expected multifidelity behavior on problems with a known analytical solution, before being applied to the reactor CFD dataset, which has no such ground truth to check against directly.

4.2. Cross-Fidelity Correlation Analysis Results

This section presents the cross-fidelity correlation analysis for the LF-HF and MF-HF assembly pairs across the full 193-assembly core. The data varies substantially between fidelities, which makes this a difficult problem for multifidelity machine learning methods. Pearson correlation is used as a metric throughout the paper, alongside the more standard mean absolute error (MAE) and root mean squared error (RMSE), because it captures how well a method tracks the overall shape of the transient in a way those other metrics do not. A method can score a high RMSE while still having a strong correlation, since the sharp flow rate peaks in this dataset are hard to capture exactly, so the predicted transient can track the shape correctly without ever reaching the full height of the peak.

Correlation across the core can be seen in Figure 4, which maps LF-HF and MF-HF correlation onto the core layout. This correlation is computed directly between the real transients recorded at each fidelity level during the original CFD simulations (Section 3.4), not between any trained model’s predictions and ground truth. No MFNN or MFKAN model is involved in this analysis. Across all 193 assemblies, the mean LF-HF correlation is only 0.003 (median 0.008) and the mean MF-HF correlation is 0.041 (median 0.053), both essentially zero. Even the single best assembly only reaches 0.444 for LF-HF and 0.409 for MF-HF. The two heatmaps show different spatial patterns, indicating that the cross-fidelity correlation is not uniformly weak everywhere: it varies by assembly and by which fidelity pair is being compared, so a location that is highly correlated LF-to-HF is not necessarily the same location that is highly correlated MF-to-HF.

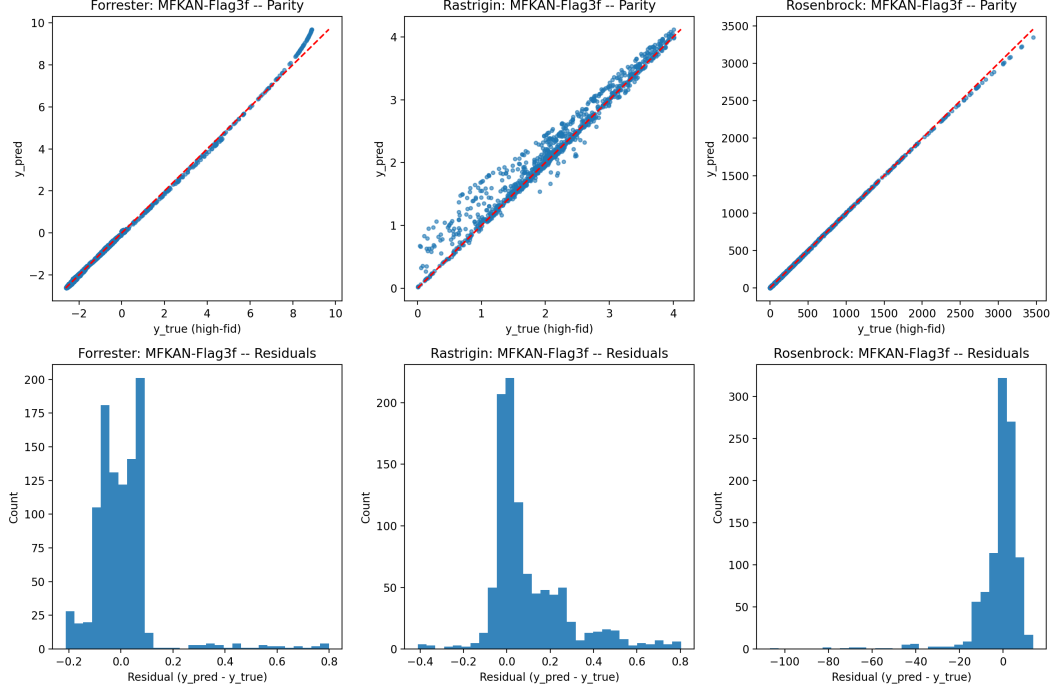


Figure 3: MFKAN-Flag3f parity plots (top) and residual histograms (bottom) on the three-fidelity Forrester (1D), Rastrigin (2D), and Rosenbrock (2D) functions.

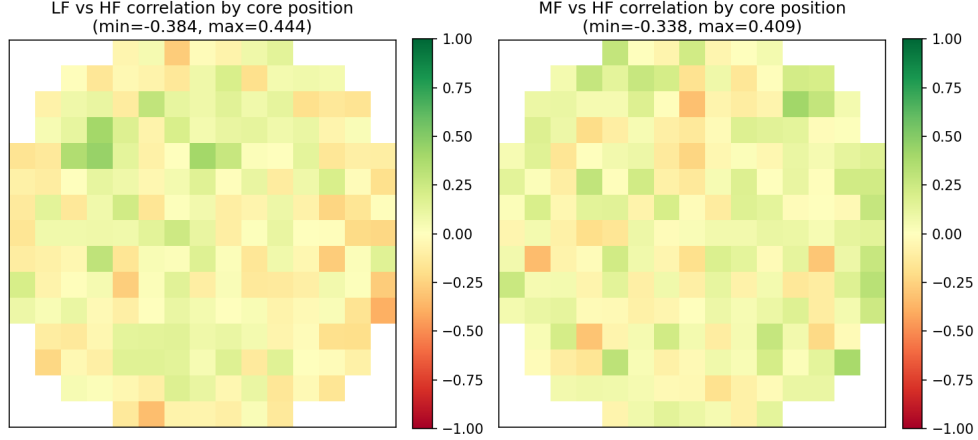


Figure 4: LF-HF and MF-HF correlation, computed between the real transients at each fidelity, mapped onto the core layout.

To illustrate the behavior of the most- and least-correlated assemblies, Figures 5 and 6 compare their mass flow rate trajectories across fidelity levels. Even the most-correlated assemblies exhibit only moderate agreement, highlighting the difficulty of this multifidelity problem. For the least-correlated assemblies, the trajectories differ substantially, providing limited cross-fidelity information for predicting the high-fidelity transient from low- or medium-fidelity data.

Overall, cross-fidelity correlation remains weak across the core, motivating the central hypothesis examined in Section 4.3: that the predictive performance of MFNN and MFKAN is fundamentally constrained by the limited correlation between the low- and medium-fidelity data and the high-fidelity ground truth.

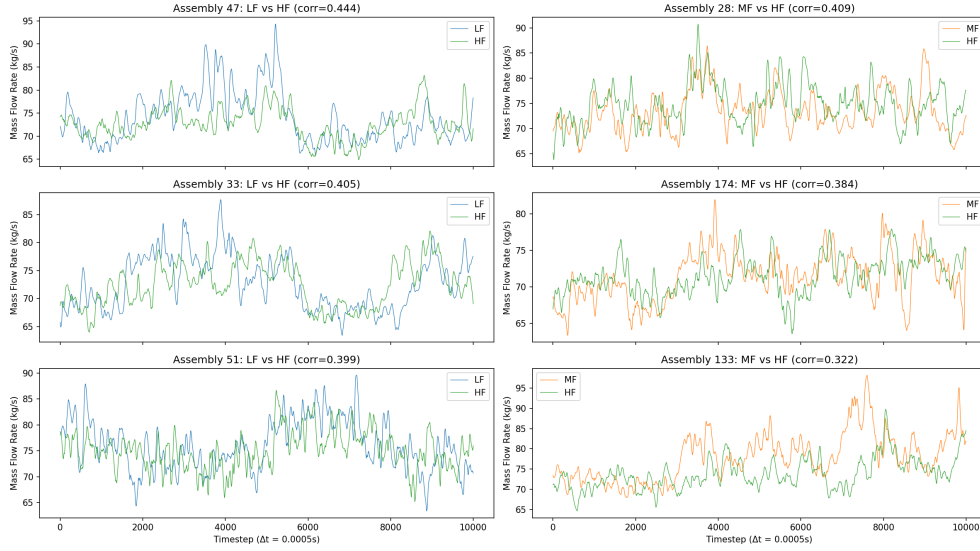


Figure 5: Mass flow rate trajectories for the **highest-correlation assemblies**, with LF–HF and MF–HF pairs ranked independently. Even the highest-correlation cases exhibit only moderate agreement.

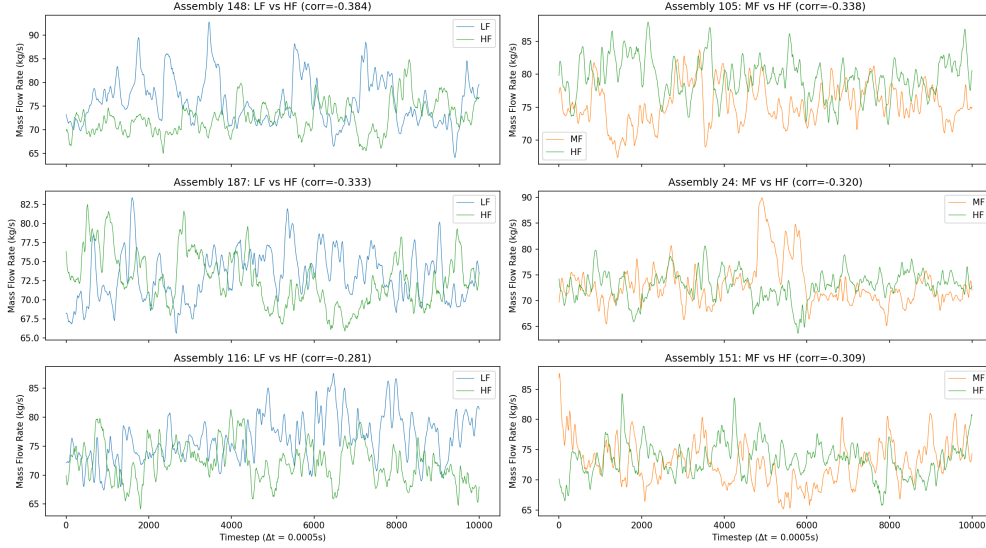


Figure 6: Mass flow rate trajectories for the **lowest-correlation assemblies**, with LF–HF and MF–HF pairs ranked independently. These cases exhibit weak cross-fidelity agreement.

4.3. One-Time-Step-Forward Prediction Results

These results correspond to the one-time-step-ahead methodology, in which all low-fidelity data (and medium-fidelity data for the three-fidelity methods) are used for training together with 20% of the high-fidelity data. The remaining 80% of the high-fidelity transient is then predicted one time step ahead and compared with the ground truth. This approach can be considered one-step-ahead forecasting because each prediction advances beyond the state provided to the model; however, the errors do not compound over time because each prediction uses the true input state rather than feeding the previous prediction back as the next input. *This differs from the TabDDPM-based forecasting approach presented in Section 4.4, where predictions are propagated forward recursively.*

Three experiments are considered in this section: a baseline using all assemblies without grouping or subsetting, a spectral-clustering approach based on Section 3.5, and a subset containing the most highly correlated assemblies.

4.3.1. Hyperparameter Tuning Results

Final tuned hyperparameters for the MFKAN and MFNN baseline methods (search space and tuning methodology in Section 3.10) are reported in Tables 2 and 3.

Table 2: MFKAN baseline hyperparameters (tuned).

Method	Depth	Hidden	Grid	k	Steps	Learning Rate	λ	$\lambda_{entropy}$	Reg. Metric
Delta	1	8	4	8	75	0.011	8.568e-04	6.7996	edge_backward
Flag	1	8	3	8	250	0.0079	7.557e-04	6.7701	node_backward
Flag3f	1	8	3	7	75	0.092	5.761e-04	4.0174	edge_forward_spline_u

Table 3: MFNN baseline hyperparameters (tuned).

Method	Hidden	Learning Rate	Epochs	Weight Decay
MFNN-Delta	[128, 128]	0.0026	4000	1.059e-06
MFNN-Flag	[16, 16, 16]	0.0093	2000	3.638e-04
MFNN-Flag3f	[16, 16, 16]	0.016	3000	3.883e-05

4.3.2. Baseline

This is the baseline case: all 193 assemblies are used together as a single group, with no clustering and no correlated-assembly subsetting. Inputs and outputs are PCA-compressed to 10 components (Section 3.3) before training the MFNNs and MFKANs.

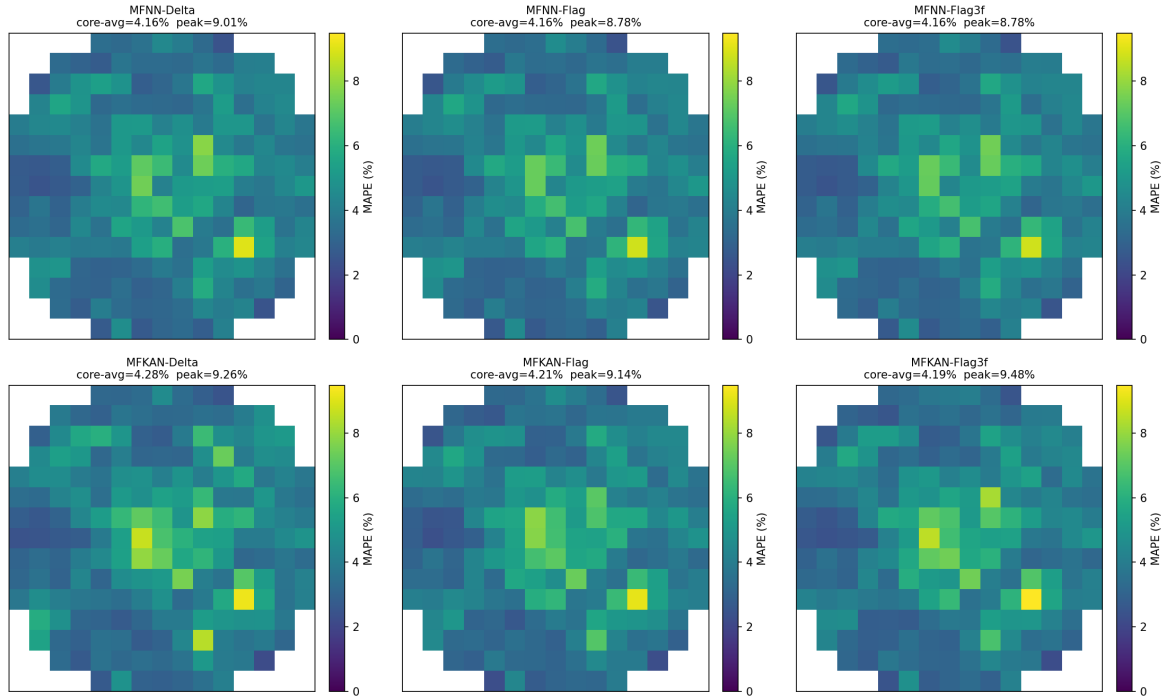


Figure 7: Assembly-wise MAPE distributions for all six multifidelity methods using 20% of the high-fidelity data for training in the **baseline** case. Core-average MAPE and peak assembly MAPE are reported for each method.

As shown in Figure 7, all six methods performed similarly, with the MFNN methods slightly outperforming the MFKAN methods. The lowest core-average MAPE is tied between MFNN-Delta, MFNN-Flag, and MFNN-Flag3f

at 4.16%, and the lowest peak MAPE is tied between MFNN-Flag and MFNN-Flag3f at 8.78%. The MFKAN methods performed similarly to the MFNN methods overall, with the added benefit of interpretability.

A constant random seed was used, but repeated runs still showed some variability, prompting the reproducibility study in Table 4. MFKAN-Delta showed the most run-to-run variability of the six methods, consistent with the random-seed sensitivity reported for KAN generally [41]; MFNN was highly stable, and MFKAN-Flag/Flag3f varied only slightly. Aside from this reproducibility study itself, a single MFKAN-Delta run is used throughout the rest of the paper, so its results may vary slightly on rerun.

Table 4: Reproducibility across repeated runs (mean $\pm$ std).

Method	N Repeats	RMSE Mean	RMSE Std	MAE Mean	MAE Std
MFKAN-Delta	5	4.3160	0.0195	3.2850	0.0145
MFKAN-Flag	5	4.2662	0.0014	3.2583	0.0018
MFKAN-Flag3f	5	4.2674	0.0026	3.2478	0.0013
MFNN-Delta	5	4.1733	0.0000	3.2162	0.0000
MFNN-Flag	5	4.1673	0.0000	3.2130	0.0000
MFNN-Flag3f	5	4.1672	0.0000	3.2130	0.0000

4.3.3. Clustered

This section repeats the six-method MFNN versus MFKAN comparison from Section 4.3.2, but first applies spectral clustering (Section 3.5) to partition the 193 assemblies into four clusters, with a separate network trained for each cluster. The same 20%/80% HF training/test split is maintained. The inputs and outputs are PCA-compressed to three components rather than the 10 used in the baseline case, reflecting the reduced dimensionality already achieved through spectral clustering. Figure 8 presents the results, showing that clustering degrades both MFNN and MFKAN performance relative to the baseline case, though MFKAN degrades noticeably less, particularly in peak MAPE – reversing the baseline ranking where MFNN had been slightly ahead. Hyperparameters were tuned only on the baseline case (Section 3.10) and reused as-is here, which likely contributes to this across-the-board drop.

4.3.4. Most Correlated

This experiment tests the hypothesis that multifidelity methods benefit from stronger correlations between fidelity levels, which is also supported by the results here. The analysis is restricted to the 50 most strongly correlated assemblies (Section 3.4.1), rather than the full 193-assembly core. Therefore, the absolute errors are not directly comparable to those of the baseline case in Section 4.3.2; instead, this experiment provides a relative assessment of how stronger cross-fidelity correlation affects multifidelity performance.

Figure 9 shows that all six methods achieve substantially lower errors for the highly correlated subset. The lowest core-average MAPE is 3.90%, achieved by both MFNN-Flag and MFNN-Flag3f, while MFNN-Delta achieves the lowest peak MAPE at 5.43%. Consistent with the baseline results, the MFNN methods outperform their MFKAN counterparts, only slightly for core-average MAPE but more substantially for peak MAPE. The improvement over the baseline case is most pronounced in peak MAPE for both architectures.

4.3.5. One-Time-Step-Forward Spline vs. Symbolic KAN

All of the MFKAN results discussed so far (Sections 4.3.2–4.3.4) use the spline form of the KAN. As we discussed before, one of the advantages of KAN over a standard neural network is that a trained spline can be converted into a symbolic equation, so this section compares that symbolic-regression equivalent against the spline it was extracted

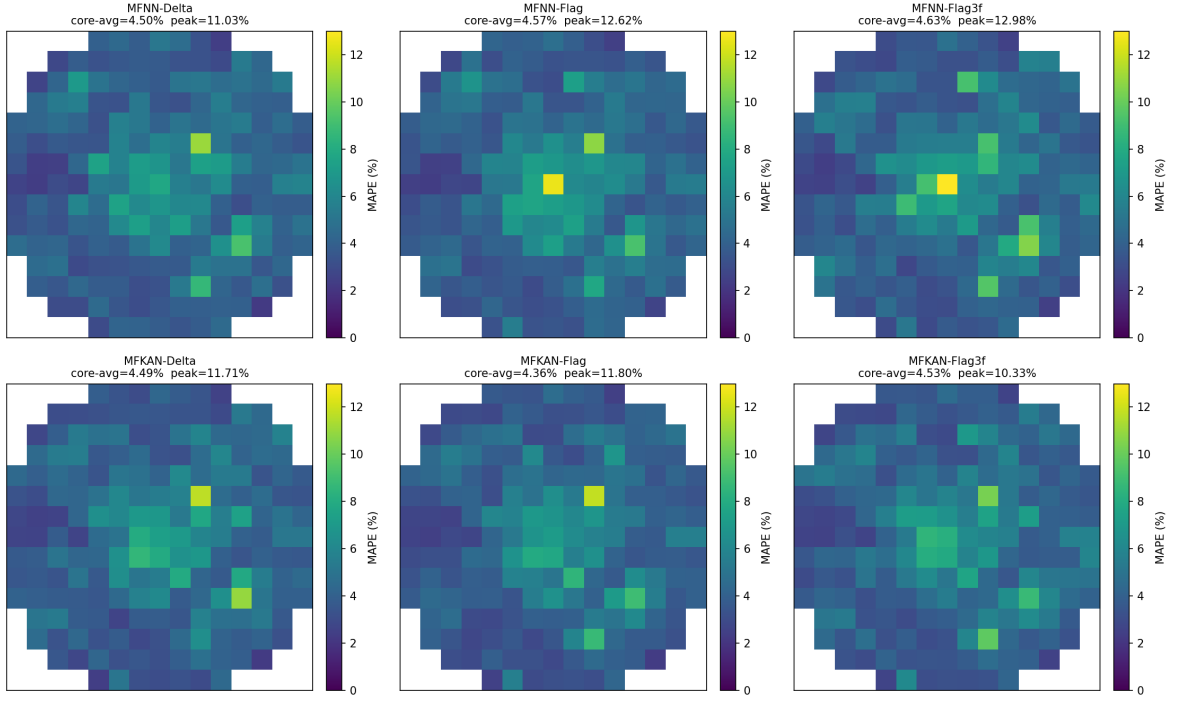


Figure 8: Assembly-wise MAPE for all six multifidelity methods using 20% of the high-fidelity data for training, with **spectral clustering** applied to partition the 193 assemblies into four clusters prior to training.

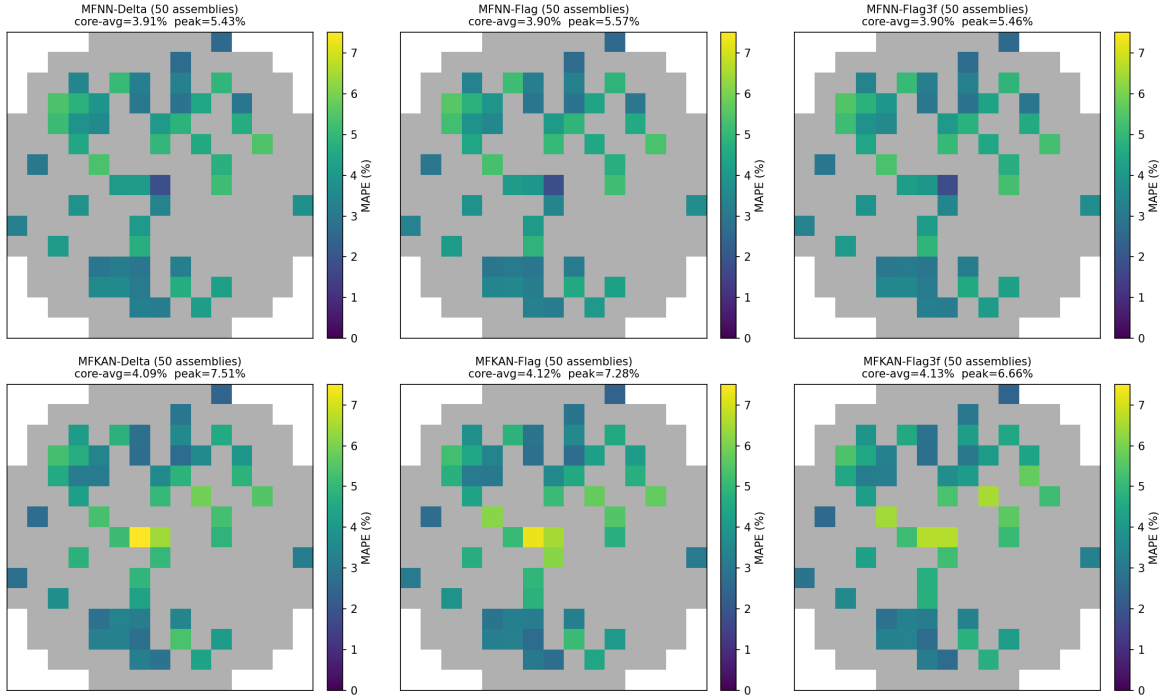


Figure 9: Assembly-wise MAPE for all six multifidelity methods using 20% of the high-fidelity data for training, evaluated on a **subset of the 50 assemblies with the highest cross-fidelity correlation**.

from, shown in Table 5. A representative example equation, extracted from MFKAN-Flag on the baseline case, is reported in Appendix A.

Symbolic regression performs comparably to the spline representation across all groupings.

Table 5: RMSE of symbolic-regression and spline representations for the MFKAN methods across the baseline, clustered, and correlated-subset cases.

Method	RMSE Spline	RMSE Symbolic
<i>Plain</i>		
MFKAN-Flag3f	4.2653	4.2707
MFKAN-Flag	4.2671	4.2689
MFKAN-Delta	4.3607	4.3484
<i>Clustered (4 binary clusters)</i>		
MFKAN-Flag3f, cluster 0	4.6772	4.5294
MFKAN-Flag3f, cluster 1	4.3026	4.1853
MFKAN-Flag3f, cluster 2	4.2055	4.0480
MFKAN-Flag3f, cluster 3	5.3492	5.2631
MFKAN-Flag, cluster 0	4.5131	4.4752
MFKAN-Flag, cluster 1	4.1843	4.1969
MFKAN-Flag, cluster 2	3.8520	3.7984
MFKAN-Flag, cluster 3	5.3098	5.3316
MFKAN-Delta, cluster 0	4.7328	4.6758
MFKAN-Delta, cluster 1	4.3067	4.2487
MFKAN-Delta, cluster 2	4.0471	3.8920
MFKAN-Delta, cluster 3	5.2682	5.2564
<i>Correlated Subset (top 50)</i>		
MFKAN-Flag3f	4.1960	4.2504
MFKAN-Flag	4.1980	4.1986
MFKAN-Delta	4.1503	4.1599

4.4. TabDDPM Synthetic Data and Forecasting

This section covers the generative models and TabDDPM-based results. Since only one real CFD transient is available at each fidelity level, synthetic data is generated with TabDDPM to give the forecasting models more to train on, and used here to run true, compounding forecasting rather than the one-time-step-forward method in Section 4.3. There are five parts to this section: how the synthetic data was generated and validated against the real transients (Section 4.4.1), hyperparameter tuning results (Section 4.4.2), the forecasting results using KAN and FNN trained on that synthetic data (Section 4.4.3), the KAN spline-vs-symbolic comparison for those same forecasts (Section 4.4.4), and an assembly-level analysis of those results (Section 4.4.5).

4.4.1. Synthetic Data Generation and Validation

Synthetic data for each assembly was generated by running TabDDPM’s generation loop, checking each new sample against every assembly’s real trajectory, and adding the sample to whichever assembly’s bucket it correlated most strongly with. This continued until every assembly had accumulated 1,000 synthetic realizations.

Figure 10 compares the synthetic and real data for a representative assembly (No. 47). Similar to the analysis in Section 4.2, correlation provides a more informative measure of agreement than error-based metrics such as RMSE.

Figure 11 presents a different measure from the cross-fidelity correlation analysis in Section 4.2. Here, the metric represents the mean correlation between the synthetic data generated for each assembly and its own real trajectory, indicating the usefulness of the synthetic data for assembly-level self-prediction and the cross-fidelity forecasting examined later in this section. TabDDPM generated strongly correlated synthetic samples more readily for some assemblies than others. For more difficult assemblies, the correlation threshold had to be progressively relaxed to obtain sufficient samples, resulting in millions of generated candidates before the required synthetic dataset was completed.

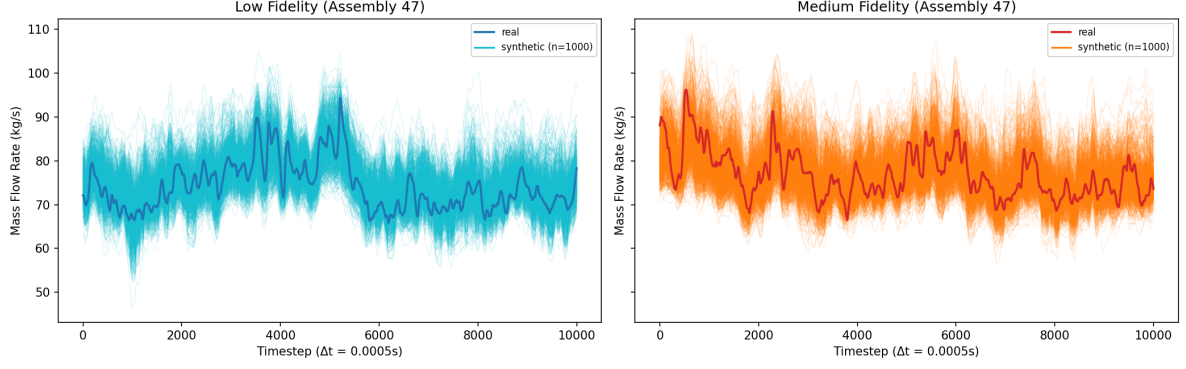


Figure 10: Real and synthetic flow rate data as a function of time steps overlaid for a selected assembly (No. 47), Low-Fidelity (left) vs. Medium-Fidelity (right).

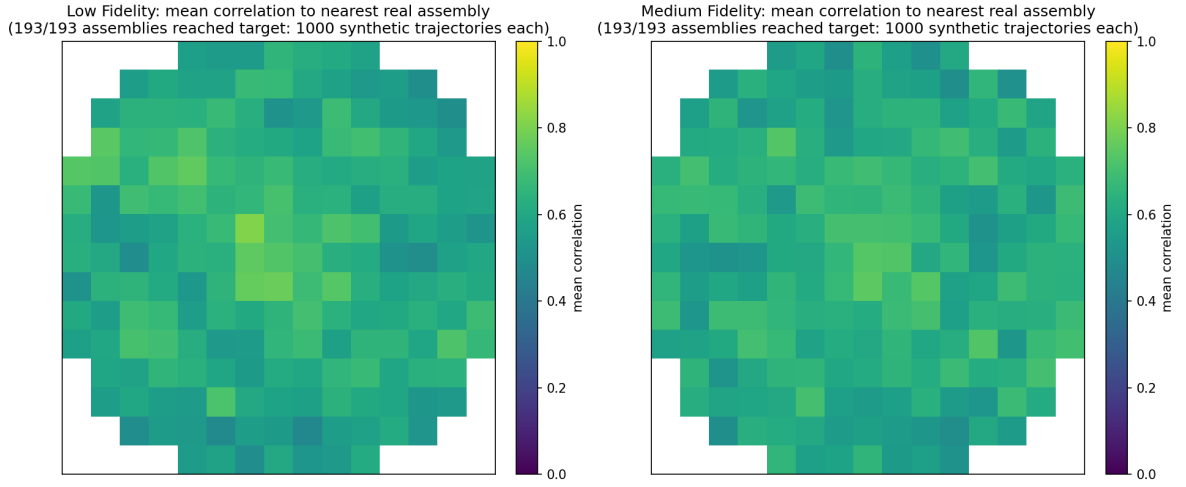


Figure 11: Whole-core mean synthetic-vs-real correlation per assembly, LF and MF side by side. (Section 3.8.1).

4.4.2. Hyperparameter Tuning Results

Final tuned hyperparameters for the dedicated-fidelity KAN and FNN forecasting models (search space and tuning methodology in Section 3.10) are reported in Tables 6 and 7; the combined (LF+MF→HF) models reuse the dedicated low-fidelity model’s values.

Table 6: KAN forecasting hyperparameters (tuned; combined reuses LF’s values – see `apply_forecasting_lf_mf.py`).

Variant	Depth	Hidden	Grid	k	Steps	λ	$\lambda_{entropy}$	Learning Rate	Reg. Metric
LF	1	32	10	8	250	6.713e-06	0.1175	0.0116	edge_forward_spline_n
MF	1	32	8	4	250	3.622e-05	4.0001	0.0299	edge_forward_sum
COMBINED	1	32	10	8	250	6.713e-06	0.1175	0.0116	edge_forward_spline_n

Table 7: FNN forecasting hyperparameters (tuned; combined reuses LF’s values – see `apply_forecasting_lf_mf.py`).

Variant	Depth	Hidden	Learning Rate	Epochs	Weight Decay
LF	4	128	0.0078	500	4.092e-06
MF	4	256	0.005	1000	1.130e-06
COMBINED	4	128	0.0078	500	4.092e-06

4.4.3. Augmented Forecasting Results

This section shows the results of forecasting using static methods, comparing KAN and FNN. These are no longer MFNN/MFKAN per our typical definition in Section 3.7. They nonetheless retain a multifidelity character, since the synthetic training data spans all three fidelity levels, despite each model being trained and evaluated using a single-fidelity approach. We first consider self-evaluation, where a model trained on synthetic data from a given fidelity predicts the corresponding real transient at that same fidelity.

Figure 12 shows that KAN and FNN achieve comparable self-evaluation fits, with neither architecture consistently capturing the sharp peaks better than the other. This is also reflected quantitatively in Table 8, where the LF→LF and MF→MF cases denote single-fidelity self-evaluation. Using only the initial mass flow rate condition ($N = 1$), KAN achieves self-evaluation correlations of 0.80–0.81 across these two cases, closely matched by FNN’s 0.78–0.81.

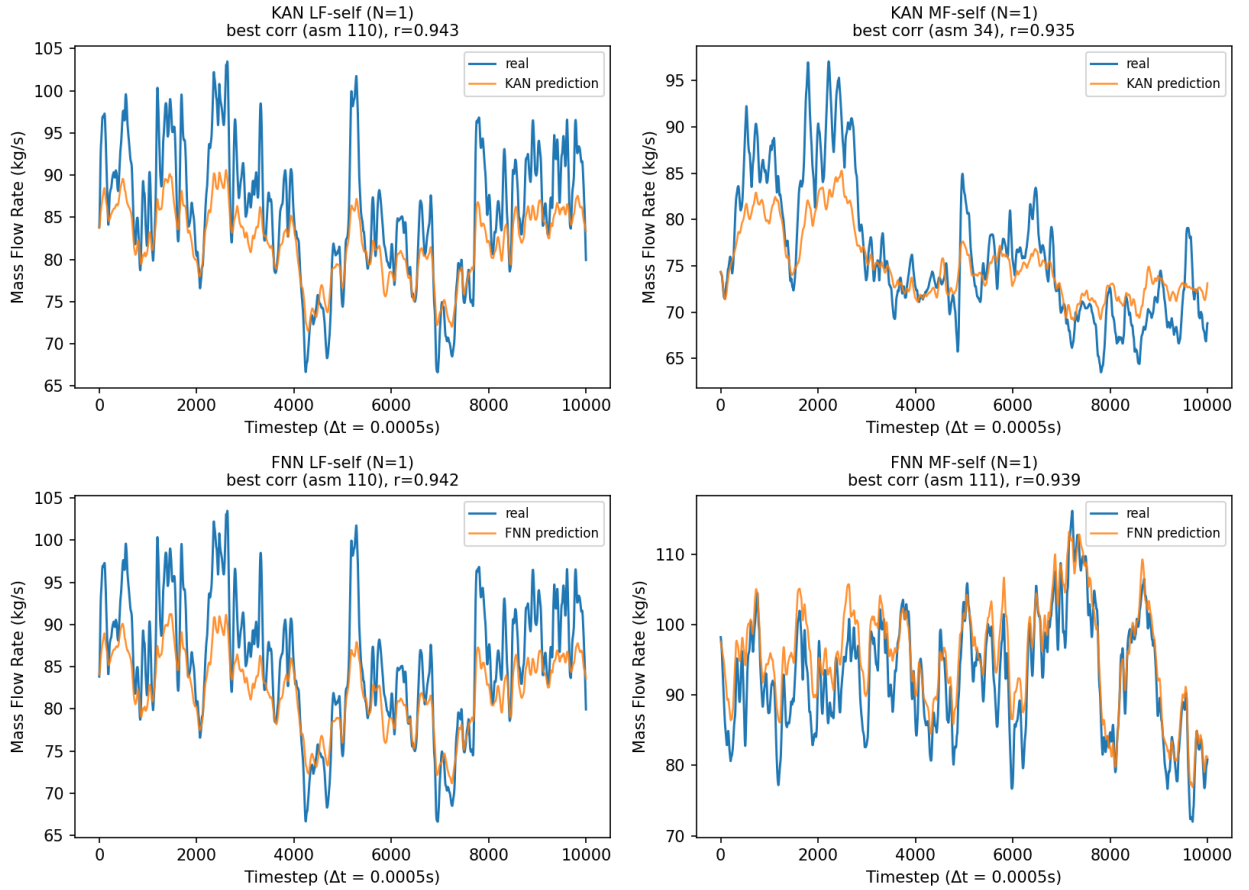


Figure 12: Self-evaluation performance of KAN and FNN using synthetic data with a single initial condition ($N = 1$). Rows correspond to the KAN (top) and FNN (bottom) architectures, while columns correspond to low-fidelity (LF-self, left) and medium-fidelity (MF-self, right) evaluation. Each panel compares the predicted and real mass flow rate transient for the assembly with the highest Pearson correlation (r), reported above the panel. The KAN model used in this figure is a spline KAN.

Figure 13 presents the cross-fidelity forecasting case, where all synthetic data for a given LF/MF combination are used to predict the real high-fidelity transient. Each panel shows the best-performing assembly for the corresponding method and fidelity direction. Even for these best-performing cases, neither method fully captures the real transient behavior. Notably, both KAN and FNN achieve their best performance for the same assembly in two of the three directions (assembly 51 for LF→HF and assembly 28 for MF→HF), further motivating an examination of whether forecasting performance is related to the cross-fidelity correlations identified in Section 4.2. This analysis will be

Table 8: TabDDPM cross-fidelity forecast: KAN vs. FNN.

Direction	N	KAN			FNN		
		Correlation	RMSE	MAE	Correlation	RMSE	MAE
LF $\rightarrow$ LF	1	0.8030	3.0319	2.4323	0.7763	3.0763	2.4587
MF $\rightarrow$ MF	1	0.8059	3.2755	2.6714	0.8120	3.1850	2.5877
LF $\rightarrow$ HF	1	0.0336	5.5506	4.5281	0.0384	5.6863	4.6463
MF $\rightarrow$ HF	1	0.0582	5.4903	4.4897	0.0592	5.5980	4.5668
LF+MF $\rightarrow$ HF	1	0.0562	5.3724	4.3844	0.0370	7.2223	5.8852
LF $\rightarrow$ LF	10	0.7327	3.4880	2.8397	0.7567	3.1896	2.5453
MF $\rightarrow$ MF	10	0.7310	3.8932	3.2192	0.7289	3.7422	3.0649
LF $\rightarrow$ HF	10	0.0345	5.6584	4.6258	0.0405	5.6006	4.5745
MF $\rightarrow$ HF	10	0.0554	5.9511	4.8955	0.0591	5.9294	4.8780
LF+MF $\rightarrow$ HF	10	0.0573	5.5368	4.5427	0.0553	8.5356	7.0185

presented later in Section 4.4.5.

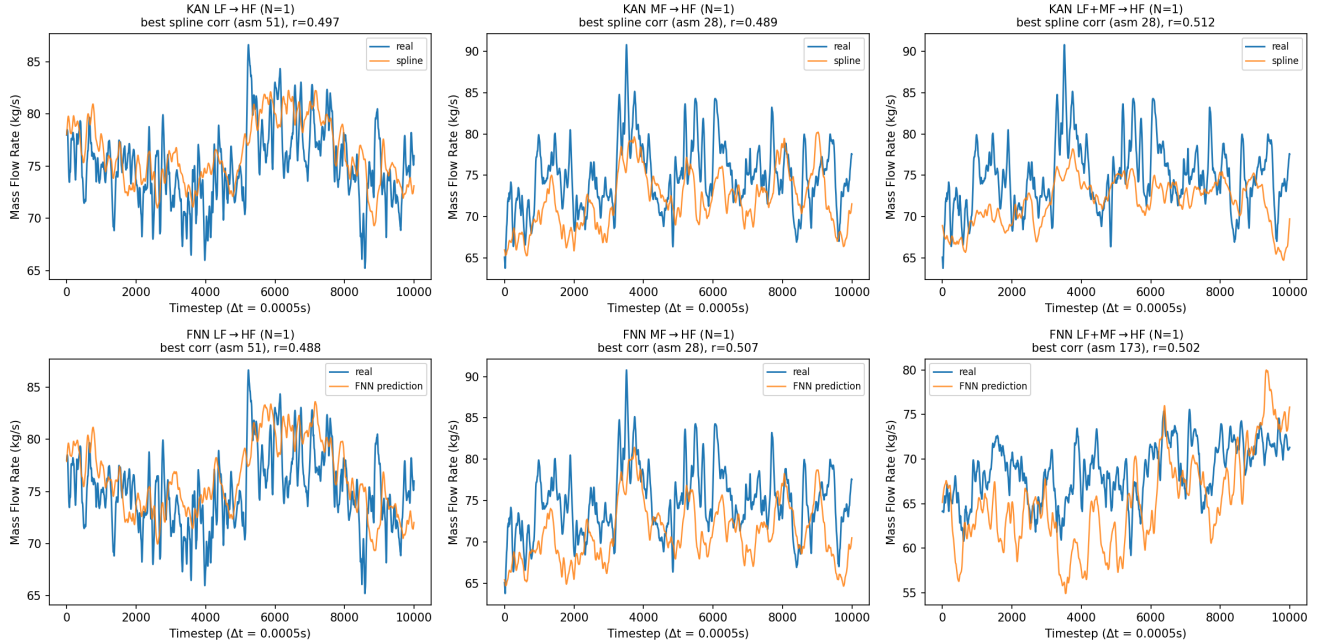


Figure 13: Cross-fidelity forecasting performance of KAN and FNN using synthetic data with a single initial condition ($N = 1$). Rows correspond to KAN (top) and FNN (bottom), while columns correspond to LF $\rightarrow$ HF (left), MF $\rightarrow$ HF (center), and combined LF+MF $\rightarrow$ HF (right) forecasting. Each panel compares the predicted and real HF mass flow rate transient for the assembly with the highest Pearson correlation (r), reported above the panel. KAN results shown here use the learned spline representation.

Table 8 reports the complete results, including the $N = 10$ case, where the first 10 mass flow rate timesteps from the real HF simulation are provided as initial conditions rather than only the first timestep. Performance degrades at $N = 10$ relative to $N = 1$, likely due to the increasing input dimensionality. KAN’s self-evaluation correlation decreases from 0.80–0.81 at $N = 1$ to 0.73 at $N = 10$, and FNN degrades similarly, from 0.78–0.81 to 0.73–0.76 – the two architectures remain comparable in self-evaluation at both N . Both methods perform poorly in cross-fidelity forecasting at either N , with correlations of only 0.03–0.06. This behavior is consistent with the weak cross-fidelity correlations identified in Section 4.2.

4.4.4. TabDDPM Spline vs. Symbolic KAN

All KAN forecasting results in Section 4.4.3 use the learned spline representation. Following the analysis in Section 4.3.5, this section compares the symbolic-regression equations extracted from the trained KANs against their original spline representations.

Table 9 shows that symbolic regression outperforms the spline representation across all six forecasting directions. The spline metrics reported here are recomputed by reloading each saved model, so they differ slightly (by roughly 6–8% in RMSE) from the corresponding rows in Table 8, reflecting minor differences between the two evaluation pipelines rather than a different underlying model. The spline fit itself is comparatively weak in this cross-fidelity forecasting setting, with RMSE ranging from 5.70 to 6.42 kg/s. Symbolic regression reduces this error in every direction, by 0.6–9.2%, with the largest reductions for the combined LF+MF→HF direction (9.2% at both $N = 1$ and $N = 10$) and for MF→HF at $N = 10$, where the spline error is highest (6.42 to 5.86 kg/s, 8.8%). The lowest symbolic error is for LF+MF→HF at $N = 1$, where RMSE drops to 5.18 kg/s from 5.70 kg/s for the spline.

Table 9: RMSE of symbolic-regression and spline representations for the TabDDPM-augmented cross-fidelity KAN forecasts.

Direction	RMSE Spline	RMSE Symbolic
LF → HF, $N = 1$	5.8893	5.8512
MF → HF, $N = 1$	5.8348	5.5607
LF+MF → HF, $N = 1$	5.6988	5.1754
LF → HF, $N = 10$	6.0100	5.7142
MF → HF, $N = 10$	6.4182	5.8559
LF+MF → HF, $N = 10$	5.9208	5.3785

4.4.5. Assembly Analysis

In Section 4.4.3, both KAN and FNN achieved their best forecast performance for the same assembly in two of the three directions: assembly 51 for LF→HF and assembly 28 for MF→HF. This observation motivates an assembly-level analysis of whether forecast performance is related to the underlying correlation between fidelity levels. Figures 14 and 15 compare these quantities for KAN and FNN, respectively. Each point represents one fuel assembly: the horizontal axis gives the real-data cross-fidelity correlation between the LF or MF transient and the corresponding HF transient, while the vertical axis gives the correlation between the model-predicted and real HF transients for that assembly. *Ideally, stronger real-data cross-fidelity correlation should correspond to forecast correlations approaching unity, producing a strong positive relationship across assemblies.* Such a relationship would indicate that assemblies with greater similarity between fidelity levels provide more transferable information for HF forecasting.

For KAN, this relationship is strong, with $r = 0.840$ for LF→HF and $r = 0.896$ for MF→HF. The relationship is similarly strong for FNN, with $r = 0.804$ for LF→HF and $r = 0.903$ for MF→HF. Thus, for both architectures, assemblies with stronger real-data cross-fidelity correlation generally exhibit higher forecast correlation, indicating that both KAN and FNN predict the HF transient more effectively when stronger transferable structure exists between fidelity levels. This provides an assembly-level explanation for the challenging full-core forecasting results in Section 4.4.3: when the underlying fidelities are weakly correlated, there is limited information from which to infer the HF transient, regardless of architecture.

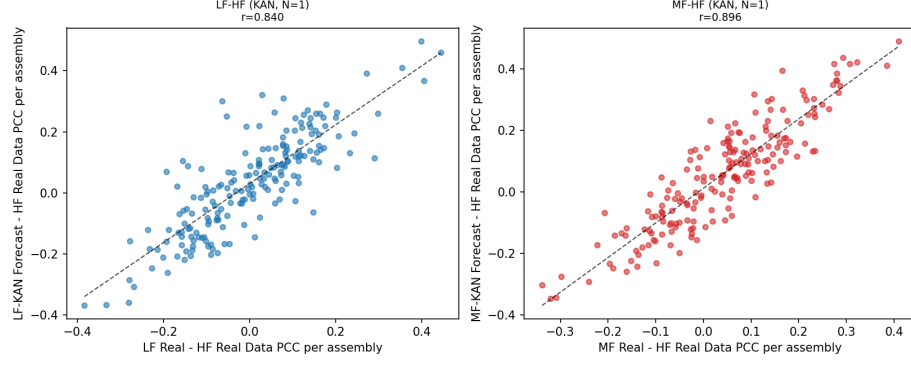


Figure 14: Assembly-level relationship between real-data cross-fidelity correlation and KAN forecasting performance for $N = 1$. Each point represents one fuel assembly. The horizontal axis gives the correlation between the real LF or MF transient and the real HF transient, while the vertical axis gives the correlation between the KAN-predicted and real HF transients. Results are shown for LF $\rightarrow$ HF (left) and MF $\rightarrow$ HF (right), with the Pearson correlation (r) between these two assembly-level quantities reported above each panel.

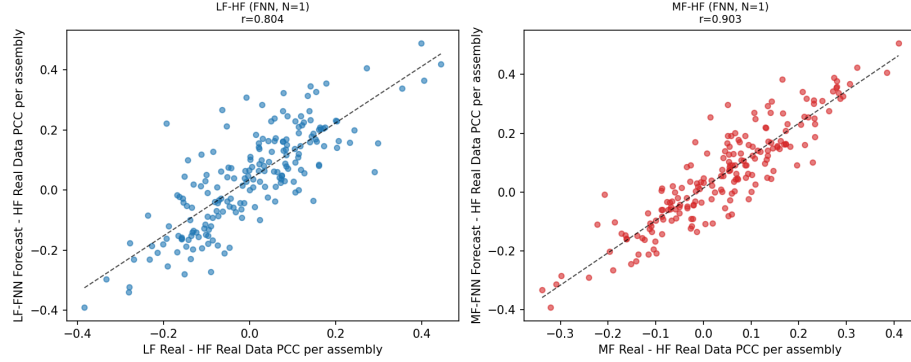


Figure 15: Assembly-level relationship between real-data cross-fidelity correlation and FNN forecasting performance for $N = 1$. Each point represents one fuel assembly. The horizontal axis gives the correlation between the real LF or MF transient and the real HF transient, while the vertical axis gives the correlation between the FNN-predicted and real HF transients. Results are shown for LF $\rightarrow$ HF (left) and MF $\rightarrow$ HF (right), with the Pearson correlation (r) between these two assembly-level quantities reported above each panel.

5. Discussion

Predicting the transient flow field considered in this work presents a particularly challenging multifidelity machine learning problem: cross-fidelity correlations are weak across most of the core (Section 4.2), the underlying transients are highly oscillatory and spatially heterogeneous, and only a single transient is available for training at each fidelity level. The central question motivating this work is whether multifidelity learning can reduce the amount of expensive high-fidelity data required to predict this transient, evaluated here using both MFNN and MFKAN, with MFKAN additionally offering interpretability through closed-form symbolic equations unavailable with MFNN if it can match MFNN’s accuracy. Neither architecture performs particularly well in absolute terms, especially for TabDDPM-augmented forecasting (Section 4.4.3). MFKAN nonetheless remains competitive with MFNN throughout. This holds for the baseline, clustered, and correlated-subset one-time-step-forward cases (Section 4.3), as well as for TabDDPM self-evaluation (correlations of roughly 0.73–0.81 for both) and true cross-fidelity forecasting.

Cross-fidelity correlation emerges as the main driver of achievable accuracy, independent of architecture. The correlated-assembly subset (Section 4.3.4) improves core-average and peak MAPE substantially over the full-core baseline for both MFNN and MFKAN, so the benefit of multifidelity learning in this problem is assembly-dependent rather than uniform across the core, concentrated in assemblies where the fidelity levels are already well correlated.

These correlated-subset results, however, have limited practical applicability on their own, since they consider only the 50 assemblies already identified as having strong cross-fidelity correlations rather than enabling full-core prediction. The assembly-level analysis in the TabDDPM setting (Section 4.4.5) supports this same correlation-driven mechanism directly: both KAN’s and FNN’s per-assembly forecast correlation track real cross-fidelity correlation comparably closely ($r = 0.840\text{--}0.896$ for KAN versus $r = 0.804\text{--}0.903$ for FNN), indicating both architectures are similarly sensitive to the physical cross-fidelity structure actually present in the data, rather than this being a KAN-specific advantage.

Clustering degrades one-time-step-forward accuracy for both MFNN and MFKAN relative to the baseline case, though MFKAN noticeably less so, reversing the baseline ranking (Section 4.3.3). This degradation is likely attributable, at least in part, to hyperparameters being tuned only on the baseline case and reused as-is for the clustered case rather than with a clustering-specific search; it may also suggest that spectral clustering is not a particularly effective preprocessing step for this problem.

All one-time-step-forward results use a fixed 20% high-fidelity training split, where the complete low- and medium-fidelity transients and the first 20% of the high-fidelity transient are available for training (Section 4.3). Additional real high-fidelity data would likely improve prediction over the remaining unseen portion. However, testing that directly would require more real HF CFD data, and that same scarcity is what the TabDDPM-based data augmentation in this work is designed to address instead (Section 4.4). Repeating this analysis across different real HF training splits was therefore not pursued.

The TabDDPM-augmented forecasts are considerably weaker than the one-time-step-forward predictions, with cross-fidelity correlations of only 0.03–0.06 (Table 8). Three factors likely explain this gap. First, the forecasting task is harder: unlike the one-time-step-forward models, the forecasting models see no real HF data during training and propagate their own predictions, so errors compound. Second, synthetic augmentation adds data quantity but not cross-fidelity information: the LF and MF samples reproduce their own fidelity well (self-evaluation correlations of 0.73–0.81) but carry no more information about HF than the real data, so forecast accuracy follows the underlying cross-fidelity correlation (Section 4.4.5). Third, all samples derive from a single real transient per fidelity level, which limits their diversity. Generative augmentation of lower-fidelity data alone is therefore not a substitute for real HF information when fidelities are weakly correlated; generative models conditioned on LF or MF inputs to produce HF-like samples are a promising alternative.

KAN’s interpretability advantage, through the symbolic regression it enables (Section 3.6.2), persists across both experimental settings. For one-time-step-forward prediction, the extracted symbolic equations match the spline representation’s accuracy rather than exceeding it (Section 4.3.5); for the TabDDPM-augmented forecasting experiments, symbolic regression instead improves on the spline representation across all six forecasting directions, most substantially for the combined LF+MF→HF direction and for MF→HF at $N = 10$, where RMSE drops from 6.42 kg/s (spline) to 5.86 kg/s (symbolic) (Section 4.4.4). Even this improved fit is low in absolute terms, reflecting how difficult the underlying forecasting task is rather than a shortcoming specific to symbolic regression. Taken together, these results indicate that MFKAN’s interpretability does not come at an additional accuracy cost beyond what FNN and the spline representation already pay in this difficult forecasting setting. This interpretability is also not limited by the PCA compression used throughout this work (Section 3.3). The equations are reported in PCA-component space (Appendix A), and recovering the physical mass flow rate from them only requires applying the PCA inverse transform, a quick linear operation also provided in the accompanying repository.

Some limitations of this study should be noted. Only a single transient is available at each fidelity level, rather than many independent realizations, which limits how far these results can be expected to generalize (Section 3.1.3). MFKAN also showed some run-to-run variance, particularly for MFKAN-Delta, and much less so for MFKAN-

Flag and MFKAN-Flag3f (Table 4); consistent with this, MFKAN-Flag generally achieved slightly better accuracy than MFKAN-Delta across the one-time-step-forward experiments, though typically by a modest margin. The TabDDPM synthetic-data validation reported in Section 4.4.1 is limited to correlation-based comparisons; a fuller statistical or physical validation of the synthetic transients has not yet been explored and may become a subject for future work.

A natural extension of this work is to move beyond the static prediction methods considered here toward a fully autoregressive, time-dependent architecture, where predictions are recursively used as inputs for subsequent timesteps rather than generated independently. Time-dependent KAN architectures have already been proposed, such as TKAN [70], which incorporates recurrent memory into the KAN framework for multi-step time-series forecasting; however, to our knowledge, such temporal KAN architectures have not yet been combined with multifidelity learning. These architectures were intentionally not considered in the present study because introducing recurrent dependencies would likely increase the complexity of the resulting symbolic expressions and diminish the interpretability that motivates the use of KAN. A future extension could therefore shift the emphasis from symbolic interpretability toward improved long-term forecasting accuracy, investigating whether temporal multifidelity KAN architectures can provide more accurate multi-step predictions despite the associated increase in model complexity.

6. Conclusions

This work evaluated whether multifidelity surrogate modeling, using three CFD fidelity levels, can reduce the amount of expensive high-fidelity data required to predict the spatiotemporal mass flow rate field across all 193 fuel assemblies of a pressurized water reactor core. Multifidelity feedforward neural networks (MFNNs) and multifidelity Kolmogorov-Arnold Networks (MFKANs) were compared for this task, with MFKAN offering interpretability as an additional benefit contingent on matching MFNN’s predictive accuracy. The problem was examined through both one-time-step-forward prediction and TabDDPM-augmented cross-fidelity forecasting under challenging conditions characterized by weak cross-fidelity correlations, highly oscillatory and spatially heterogeneous transients, and only one available CFD transient per fidelity level. For one-time-step-forward prediction, MFKAN remained competitive with MFNN across the baseline, clustered, and highly correlated assembly cases, while the strong improvement observed for the correlated subset demonstrated the importance of cross-fidelity similarity for effective multifidelity learning. In the synthetic-data experiments, KAN and FNN achieved comparable self-fidelity forecasting accuracy (correlations of roughly 0.73–0.81 for both), and both architectures struggled with true cross-fidelity forecasting. Assembly-level analysis further showed that both KAN’s and FNN’s forecasting performance was strongly associated with the underlying cross-fidelity correlation, to a comparable degree for both architectures. Beyond predictive performance, MFKAN provides an important advantage over MFNN through its ability to extract closed-form symbolic representations. These equations matched or exceeded the accuracy of their spline counterparts in both the one-time-step-forward and TabDDPM-augmented settings, though absolute accuracy in the latter setting remained modest for both representations, with symbolic RMSE of 5.18–5.86 kg/s across the six forecasting directions. Overall, these results show that KAN-based multifidelity modeling achieves predictive accuracy comparable to conventional neural-network approaches, while additionally offering interpretability unavailable from MFNN, even though both architectures’ absolute accuracy was often modest in the more difficult forecasting settings. Cross-fidelity correlation and the limited availability of transient data, rather than any inherent limitation of either architecture, emerge as the primary constraints on this full-core multifidelity forecasting problem.

Data Availability

The authors have all the data and codes to reproduce all the results in this work currently in a private GitHub repository. To ensure the confidentiality of this research, the authors will make this repository public during an advanced stage of the review process, which will be listed under our research group’s public Github page: <https://github.com/aims-umich>

Acknowledgment

This research is sponsored by the Department of Energy Office of Nuclear Energy through the Nuclear Energy University Programs (Award number: DE-NE0009506). The authors would like to thank Nataly Panczyk and Hyungjun Kim for their assistance with initial code implementation and for reviewing and editing this manuscript.

AI Usage Statement

The authors used Claude and Claude Code (Anthropic) as assistive tools in this work for manuscript writing, including language editing, grammar correction, and clarity improvement; literature assistance, including citation verification against original sources and search-term generation; coding, including code generation, debugging, and documentation for data processing, figure and table generation, and paper-asset pipeline scripts; data analysis, including computing and verifying summary statistics reported in the Results section; and visualization, including plotting scripts and figure generation. All cited references were independently located and verified, all AI-assisted code was reviewed and tested, numerical results were independently checked, and the underlying data, scientific content, interpretations, arguments, and conclusions were developed and verified by the authors. *All AI-generated or AI-modified material was reviewed and, where applicable, independently tested or validated, and the authors take full responsibility for the final content and conclusions.*

CRedit Author Statement

- **Meredith Eaheart:** Conceptualization, Methodology, Software, Formal Analysis, Visualization, Investigation, Data Curation, Writing - Original Draft.
- **Majdi I. Radaideh:** Conceptualization, Methodology, Software, Validation, Data Curation, Funding Acquisition, Supervision, Project Administration, Writing - Review and Edit.

References

- [1] D. Price, N. Roskoff, M. I. Radaideh, B. Kochunas, Thermal modeling of an eVinci™-like heat pipe microreactor using OpenFOAM, *Nuclear Engineering and Design* 415 (2023) 112709. [doi:10.1016/j.nucengdes.2023.112709](https://doi.org/10.1016/j.nucengdes.2023.112709).
- [2] L. A. Burnett, H. Kim, H.-C. Chou, A. Witoelar, R. A. Brewster, B. Forget, E. Baglietto, M. I. Radaideh, High-fidelity modeling of pressurized water reactor flow fields for data-driven applications, *Energy* (2026) 142104.
- [3] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, *Nature Machine Intelligence* 1 (5) (2019) 206–215. [doi:10.1038/s42256-019-0048-x](https://doi.org/10.1038/s42256-019-0048-x).
- [4] M. I. Radaideh, T. Kozlowski, Combining simulations and data with deep learning and uncertainty quantification for advanced energy modeling, *International Journal of Energy Research* 43 (14) (2019) 7866–7890. [doi:10.1002/er.4698](https://doi.org/10.1002/er.4698).
- [5] M. I. Radaideh, T. Kozlowski, Surrogate modeling of advanced computer simulations using deep Gaussian processes, *Reliability Engineering & System Safety* 195 (2020) 106731. [doi:10.1016/j.res.s.2019.106731](https://doi.org/10.1016/j.res.s.2019.106731).

- [6] P. A. Myers, N. Panczyk, S. Chidige, C. Craig, J. Cooper, V. Joynt, M. I. Radaideh, pyMAISE: A Python platform for automatic machine learning and accelerated development for nuclear power applications, *Progress in Nuclear Energy* 180 (2025) 105568. doi:10.1016/j.pnucene.2024.105568.
- [7] M. I. Radaideh, H. Tran, L. Lin, H. Jiang, D. Winder, S. Gorti, G. Zhang, J. Mach, S. Cousineau, Model calibration of the liquid mercury spallation target using evolutionary neural networks and sparse polynomial expansions, *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 525 (2022) 41–54.
- [8] O. F. Erdem, D. P. Broughton, J. Svoboda, C. Huang, M. I. Radaideh, Surrogate-driven design optimization with uncertainty constraints in Monte Carlo simulations, *Energy and AI* 22 (2025) 100655. doi:10.1016/j.egyai.2025.100655.
- [9] V. Sobes, B. Hiscox, E. Popov, R. Archibald, C. Hauck, B. Betzler, K. Terrani, Ai-based design of a nuclear reactor core, *Scientific reports* 11 (1) (2021) 19646.
- [10] M. I. Radaideh, K. Shirvan, PESA: Prioritized experience replay for parallel hybrid evolutionary and swarm algorithms – application to nuclear fuel, *Nuclear Engineering and Technology* 54 (10) (2022) 3864–3877.
- [11] D. Price, M. I. Radaideh, B. Kochunas, Multiobjective optimization of nuclear microreactor reactivity control system operation with swarm and evolutionary algorithms, *Nuclear Engineering and Design* 393 (2022) 111776.
- [12] M. I. Radaideh, K. Du, P. Seurin, D. Seyler, X. Gu, H. Wang, K. Shirvan, Neorl: Neuroevolution optimization with reinforcement learning, *arXiv preprint arXiv:2112.07057*.
- [13] M. G. Fernández-Godino, Review of multi-fidelity models, *arXiv preprint arXiv:1609.07196*.
- [14] B. Peherstorfer, K. Willcox, M. Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization, *Siam Review* 60 (3) (2018) 550–591.
- [15] X. Meng, G. E. Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems, *Journal of Computational Physics* 401 (2020) 109020.
- [16] M. Guo, A. Manzoni, M. Amendt, P. Conti, J. S. Hesthaven, Multi-fidelity regression using artificial neural networks: efficient approximation of parameter-dependent output quantities, *Computer methods in applied mechanics and engineering* 389 (2022) 114378.
- [17] J. Yi, J. Cheng, M. A. Bessa, Practical multi-fidelity machine learning: fusion of deterministic and bayesian models, *arXiv preprint arXiv:2407.15110*.
- [18] M. C. Kennedy, A. O’Hagan, Predicting the output from a complex computer code when fast approximations are available, *Biometrika* 87 (1) (2000) 1–13.
- [19] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, G. E. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473 (2198) (2017) 20160751.
- [20] M. Eaheart, M. I. Radaideh, Multifidelity surrogate modeling of onset of natural circulation following a depressurized loss of forced cooling in high-temperature gas reactors, *Applied Thermal Engineering* (2026) 132489.
- [21] F. L. Pena, V. D. Casás, A. Gosset, R. Duro, A surrogate method based on the enhancement of low fidelity computational fluid dynamics approximations by artificial neural networks, *Computers & fluids* 58 (2012) 112–119.
- [22] J. Li, Y. Li, T. Liu, D. Zhang, Y. Xie, Multi-fidelity graph neural network for flow field data fusion of turbomachinery, *Energy* 285 (2023) 129405.
- [23] S. Ghosh, S. Mondal, J. S. Kapat, A. Ray, Parametric shape optimization of pin fin arrays using a multi-fidelity surrogate model based bayesian method, *Applied Thermal Engineering* 247 (2024) 122876.
- [24] B. N. Hanna, N. T. Dinh, R. W. Youngblood, I. A. Bolotnov, Machine-learning based error prediction approach for coarse-grid computational fluid dynamics (cg-cfd), *Progress in Nuclear Energy* 118 (2020) 103140.
- [25] M. Oktavian, J. Nistor, J. Gruenwald, Y. Xu, Preliminary development of machine learning-based error correction model for low-fidelity reactor physics simulation, *Annals of Nuclear Energy* 187 (2023) 109788.
- [26] D. J. Toal, Some considerations regarding the use of multi-fidelity kriging in the construction of surrogate models, *Structural and Multidisciplinary Optimization* 51 (6) (2015) 1223–1245.

- [27] M. G. Fernández-Godino, C. Park, N. H. Kim, R. T. Haftka, Issues in deciding whether to use multifidelity surrogates, *AIAA Journal* 57 (5) (2019) 2039–2051. [doi:10.2514/1.J057750](https://doi.org/10.2514/1.J057750).
- [28] U. Rohde, T. Höhne, S. Kliem, B. Hemström, M. Scheuerer, T. Toppila, A. Aszodi, I. Boros, I. Farkas, P. Muehlbauer, V. Vyskocil, J. Klepac, J. Remis, T. Dury, Fluid mixing and flow distribution in a primary circuit of a nuclear pressurized water reactor—validation of CFD codes, *Nuclear Engineering and Design* 237 (15–17) (2007) 1639–1655. [doi:10.1016/j.nucengdes.2007.03.015](https://doi.org/10.1016/j.nucengdes.2007.03.015).
- [29] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, *Annual Review of Fluid Mechanics* 52 (2020) 477–508. [doi:10.1146/annurev-fluid-010719-060214](https://doi.org/10.1146/annurev-fluid-010719-060214).
- [30] H. F. S. Lui, W. R. Wolf, Construction of reduced-order models for fluid flows using deep feedforward neural networks, *Journal of Fluid Mechanics* 872 (2019) 963–994. [doi:10.1017/jfm.2019.358](https://doi.org/10.1017/jfm.2019.358).
- [31] K. Hasegawa, K. Fukami, T. Murata, K. Fukagata, CNN-LSTM based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different Reynolds numbers, *Fluid Dynamics Research* 52 (6) (2020) 065501. [doi:10.1088/1873-7005/abb91d](https://doi.org/10.1088/1873-7005/abb91d).
- [32] M. I. Radaideh, C. Pigg, T. Kozlowski, Y. Deng, A. Qu, Neural-based time series forecasting of loss of coolant accidents in nuclear power plants, *Expert Systems with Applications* 160 (2020) 113699.
- [33] M. I. Radaideh, C. Pappas, J. Walden, D. Lu, L. Vidyaratne, T. Britton, K. Rajput, M. Schram, S. Cousineau, Time series anomaly detection in power electronics signals with recurrent and ConvLSTM autoencoders, *Digital Signal Processing* 130 (2022) 103704. [doi:10.1016/j.dsp.2022.103704](https://doi.org/10.1016/j.dsp.2022.103704).
- [34] R. A. Saleem, M. I. Radaideh, T. Kozlowski, Application of deep neural networks for high-dimensional large bwr core neutronics, *Nuclear Engineering and Technology* 52 (12) (2020) 2709–2716.
- [35] H. Gong, S. Cheng, Z. Chen, Q. Li, Data-enabled physics-informed machine learning for reduced-order modeling digital twin: Application to nuclear reactor physics, *Nuclear Science and Engineering* 196 (6) (2022) 668–693. [doi:10.1080/00295639.2021.2014752](https://doi.org/10.1080/00295639.2021.2014752).
- [36] D. Price, M. I. Radaideh, B. Kochunas, Simplified matching pursuits applied to 3D nuclear reactor temperature distribution construction, *Applied Mathematical Modelling* 131 (2024) 134–158.
- [37] N. R. Panczyk, O. F. Erdem, M. I. Radaideh, Opening the ai black-box: Symbolic regression with kolmogorov-arnold networks for advanced energy applications, *Energy and AI* (2025) 100595.
- [38] M. I. Radaideh, T. Kozlowski, Analyzing nuclear reactor simulation data and uncertainty with the group method of data handling, *Nuclear Engineering and Technology* 52 (2) (2020) 287–295.
- [39] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, *Proceedings of the National Academy of Sciences* 113 (15) (2016) 3932–3937. [doi:10.1073/pnas.1517384113](https://doi.org/10.1073/pnas.1517384113).
- [40] S.-M. Udrescu, M. Tegmark, AI Feynman: A physics-inspired method for symbolic regression, *Science Advances* 6 (16) (2020) eaay2631. [doi:10.1126/sciadv.aay2631](https://doi.org/10.1126/sciadv.aay2631).
- [41] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Hou, M. Tegmark, Kan: Kolmogorov–arnold networks, in: *International conference on learning representations*, Vol. 2025, 2025, pp. 70367–70413.
- [42] S. Wang, J. Chen, X. Liu, T. Zhang, X. Chai, Q. Lu, D. Shen, H. He, Kriging to kolmogorov-arnold network model accelerated discovery of oxygen control strategy in lead-based fast reactors, *Nature Communications* 16 (1) (2025) 9774.
- [43] D. Kalesh, T. Merembayev, S. Omirbekov, Y. Amanbek, Physics informed kolmogorov-arnold network for two-phase flow model with experimental data, in: *International Conference on Computational Science and Its Applications*, Springer, 2025, pp. 447–460.
- [44] N. Kalia, R. McConkey, E. Yee, F.-S. Lien, Kolmogorov–arnold networks for turbulence anisotropy mapping, *Physics of Fluids* 37 (8).
- [45] Y. S. Ju, Uncertainty-aware flow field reconstruction using svgp kolmogorov-arnold networks, *AI Thermal Fluids* (2026) 100034.
- [46] A. A. Howard, B. Jacob, P. Stinis, Multifidelity kolmogorov–arnold networks, *Machine Learning: Science and Technology* 6 (3) (2025) 035038.

- [47] Z. Liu, M. Hu, X. Song, C. Zhang, D. Tao, Symbolic regression based on kolmogorov-arnold networks for multi-fidelity data fusion, *Information Fusion* (2026) 104582.
- [48] K. Shukla, J. D. Toscano, Z. Wang, Z. Zou, G. E. Karniadakis, A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks, *Computer Methods in Applied Mechanics and Engineering* 431 (2024) 117290. doi:10.1016/j.cma.2024.117290.
- [49] L. Xu, M. Skoularidou, A. Cuesta-Infante, K. Veeramachaneni, Modeling tabular data using conditional GAN, in: *Advances in Neural Information Processing Systems*, Vol. 32, 2019, pp. 7335–7345.
- [50] J. Yoon, D. Jarrett, M. van der Schaar, Time-series generative adversarial networks, in: *Advances in Neural Information Processing Systems*, Vol. 32, 2019, pp. 5508–5518.
- [51] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, in: *Advances in Neural Information Processing Systems*, Vol. 33, 2020, pp. 6840–6851.
- [52] A. Kotelnikov, D. Baranchuk, I. Rubachev, A. Babenko, Tabddpm: Modelling tabular data with diffusion models, in: *International conference on machine learning*, PMLR, 2023, pp. 17564–17579.
- [53] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, H. Xu, Time series data augmentation for deep learning: A survey, in: Z.-H. Zhou (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 4653–4660. doi:10.24963/ijcai.2021/631.
- [54] U. M. Nabila, L. Lin, X. Zhao, W. L. Gurecky, P. Ramuhalli, M. I. Radaideh, Data efficiency assessment of generative adversarial networks in energy applications, *Energy and AI* 20 (2025) 100501.
- [55] Y. Alanazi, M. Schram, K. Rajput, S. Goldenberg, L. Vidyaratne, C. Pappas, M. I. Radaideh, D. Lu, P. Ramuhalli, S. Cousineau, Multi-module-based CVAE to predict HVCM faults in the SNS accelerator, *Machine Learning with Applications* 13 (2023) 100484. doi:10.1016/j.mlwa.2023.100484.
- [56] L. A. Burnett, U. M. Nabila, M. I. Radaideh, Variational digital twins, *Energy and AI* 24 (2026) 100756. doi:10.1016/j.egyai.2026.100756.
- [57] U. M. Nabila, P. Seurin, L. Lin, M. I. Radaideh, Physics-informed digital twin development of thermal energy distribution systems with active learning, *Applied Energy* 416 (2026) 127903.
- [58] N. Horelik, B. Herman, B. Forget, K. Smith, Benchmark for evaluation and validation of reactor simulations (beavrs), v1. 0.1, in: *Proc. Int. Conf. Mathematics and Computational Methods Applied to Nuc. Sci. & Eng*, Vol. 7, 2013, pp. 63–68.
- [59] J. Benesty, J. Chen, Y. Huang, I. Cohen, Pearson correlation coefficient, in: *Noise reduction in speech processing*, Springer, 2009, pp. 1–4.
- [60] U. Von Luxburg, A tutorial on spectral clustering, *Statistics and computing* 17 (4) (2007) 395–416.
- [61] A. Y. Ng, M. I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: *Advances in Neural Information Processing Systems*, Vol. 14, 2002, pp. 849–856.
- [62] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, *Deep learning*, Vol. 1, MIT press Cambridge, 2016.
- [63] A. I. Forrester, A. Söbester, A. J. Keane, Multi-fidelity optimization via surrogate modelling, *Proceedings of the royal society a: mathematical, physical and engineering sciences* 463 (2088) (2007) 3251–3269.
- [64] S. van Rijn, S. Schmitt, Mf2: A collection of multi-fidelity benchmark functions in python, *Journal of Open Source Software* 5 (52) (2020) 2049.
- [65] L. Mainini, A. Serani, M. P. Rumpfkeil, E. Minisci, D. Quagliarella, H. Pehlivan, S. Yildiz, S. Ficini, R. Pellegrini, F. Di Fiore, et al., Analytical benchmark problems for multifidelity optimization methods, *arXiv preprint arXiv:2204.07867*.
- [66] D. E. Bryson, M. P. Rumpfkeil, Variable-fidelity surrogate modeling of lambda wing transonic aerodynamic performance, in: *54th AIAA Aerospace Sciences Meeting*, 2016, p. 0294.
- [67] S. Ficini, U. Iemma, R. Pellegrini, A. Serani, M. Diez, Assessing the performance of an adaptive multi-fidelity gaussian process with noisy training data: A statistical analysis, in: *AIAA AVIATION 2021 FORUM*, 2021, p. 3098.

- [68] H. Wang, Y. Jin, J. Doherty, A generic test suite for evolutionary multifidelity optimization, *IEEE Transactions on Evolutionary Computation* 22 (6) (2017) 836–850.
- [69] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.
- [70] R. Genet, H. Inzirillo, [Tkan: Temporal kolmogorov-arnold networks](#) (2025). [arXiv:2405.07344](#).
URL <https://arxiv.org/abs/2405.07344>

Appendix A. Example Symbolic Equation

Every trained KAN method in this work produces its own closed-form symbolic equation (Section 3.6.2), and reporting every one explicitly is not practical given the number of method, assembly-grouping, and fidelity combinations considered. As a representative example, this appendix reports the equation extracted from MFKAN-Flag trained on the baseline, full-core case (Section 4.3.2). The full set of extracted equations for every method and experiment, along with the PCA components, scalars, and the exact transform needed to evaluate them on raw physical inputs, is available in the group GitHub repository (Data Availability).

The variables $x_1, \dots, x_{10}$ denote the 10 PCA-compressed input components (Section 3.3); Y_{PC1} through Y_{PC10} denote the 10 PCA-compressed output components. Recovering the physical mass flow rate at each of the 193 assemblies requires inverting the PCA transform, detailed in the repository alongside each equation.

```
1 Y_PC1 = -9.2741*exp(-0.0946*x_1 - 2.1883*cos(0.2572*x_3 + 0.7199))/(3.4453*x_4 + 7.3027)
    **0.805 + 0.3629*cos(0.7525*x_9 + 8.1702) - 13.4042*cos(0.0952*x_1 + 0.0358*x_9 -
    0.2571*log(7.2523 - 3.9202*x_8) + 0.0601*tanh(0.9209*x_3 - 0.9073) + 3.2689) - 10.7336
2 Y_PC2 = 0.9787*x_2 + 0.0121
3 Y_PC3 = 2.9528*sin(0.3538*x_3 + 6.5209) - 0.6739
4 Y_PC4 = -1.3144*sqrt(4.4384 - 1.9573*x_3) + 3.8869*sin(0.0433*x_1 + 0.3688*log(3.4453*x_4 +
    7.3027) + 1.0027*cos(0.2572*x_3 + 0.7199) - 7.8439) + 3.2508
5 Y_PC5 = -6.6252*cos(0.1643*x_5 - 4.4358) - 1.8247
6 Y_PC6 = -3.7527*sin(-0.2771*x_6 + 0.0154*sin(1.0028*x_1 - 7.7918) + 0.0329*sin(0.9741*x_3 -
    3.2123) + 5.9715) + 0.0616*sin(0.1934*x_1 + 0.0727*x_9 - 0.522*log(7.2523 - 3.9202*x_8)
    + 0.122*tanh(0.9209*x_3 - 0.9073) - 4.0911) + 0.1294*cos(0.8587*x_3 + 2.0466) - 1.1524
7 Y_PC7 = 1.0048*sin(0.139*x_1 + 1.1828*log(3.4453*x_4 + 7.3027) + 3.2154*cos(0.2572*x_3 +
    0.7199) - 5.548) - 0.6751*cos(0.8276*x_3 + 8.0656) + 1.978*cos(0.2054*x_5 - 2.47) -
    1.7418*cos(0.1821*x_1 + 0.0685*x_9 - 0.4916*log(7.2523 - 3.9202*x_8) + 0.1149*tanh
    (0.9209*x_3 - 0.9073) + 4.8292) + 0.2833
8 Y_PC8 = -4.5294*sin(0.2598*x_6 - 0.0144*sin(1.0028*x_1 - 7.7918) - 0.0309*sin(0.9741*x_3 -
    3.2123) + 4.6926) + 2.1318*sin(0.2669*x_1 + 0.1004*x_9 - 0.7205*log(7.2523 - 3.9202*x_8)
    + 0.1683*tanh(0.9209*x_3 - 0.9073) - 5.4731) + 0.5057*cos(0.7073*x_9 + 8.8228) - 3.6405
9 Y_PC9 = -2.218*sin(0.4388*x_9 + 9.4949) - 0.1349
10 Y_PC10 = 0.9887*x_10 - 0.0022
```