xFODE+: Explainable Type-2 Fuzzy Additive ODEs for Uncertainty Quantification

Ertuğrul Keçeci and Tufan Kumbasar

***Abstract*—Recent advances in Deep Learning (DL) have boosted data-driven System Identification (SysID), but reliable use requires Uncertainty Quantification (UQ) alongside accurate predictions. Although UQ-capable models such as Fuzzy ODE (FODE) can produce Prediction Intervals (PIs), they offer limited interpretability. We introduce Explainable Type-2 Fuzzy Additive ODEs for UQ (xFODE+), an interpretable SysID model which produces PIs alongside point predictions while retaining physically meaningful incremental states. xFODE+ implements each fuzzy additive model with Interval Type-2 Fuzzy Logic Systems (IT2-FLSs) and constraints membership functions to the activation of two neighboring rules, limiting overlap and keeping inference locally transparent. The type-reduced sets produced by the IT2-FLSs are aggregated to construct the state update together with the PIs. The model is trained in a DL framework via a composite loss that jointly optimizes prediction accuracy and PI quality. Results on benchmark SysID datasets show that xFODE+ matches FODE in PI quality and achieves comparable accuracy, while providing interpretability.**

***Index Terms*—Type-2 Fuzzy Systems, Uncertainty Quantification, System Identification, Interpretability, Additive Models.**

I. Introduction

System Identification (SysID) has increasingly adopted Deep Learning (DL) to better represent nonlinearities [1]–[3]. A notable approach is Neural Ordinary Differential Equations (NODEs), where Neural Networks (NNs) define state derivatives [4], [5]. Several extensions have enhanced the efficiency and interpretability of the NODE [5]–[7].

Fuzzy Ordinary Differential Equations (FODEs) replace NNs in NODEs with Fuzzy Logic Systems (FLSs) to improve interpretability [8]. Yet, their rules are learned over high-dimensional antecedent spaces, and the lack of explicit Partitioning Strategies (PSs) leads to highly overlapping Gaussian Fuzzy Sets (FSs). As a result, the interpretability of FLSs is largely undermined, and FODEs effectively behave as black-box models. Beyond rule-level interpretability, both NODEs and FODEs face additional challenges arising from state representation. Because states are often unmeasured and must be estimated [1], [2], the learned representations may lack clear physical meaning. Furthermore, both frameworks offer little insight into the role of each state dimension in the dynamics. Additive modeling structures can improve interpretability by isolating each dimension's effect without giving up flexibility [9]–[11]. Explainable FODE (xFODE) [12] addresses interpretability challenges of NODE/FODE by using physically meaningful states, an additive modeling for state derivatives, and PSs in the antecedent space. However, interpretability alone is insufficient for reliability, motivating the incorporation of Uncertainty Quantification (UQ).

This work was supported by MathWorks® in part by a Research Grant. awarded to T. Kumbasar. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of MathWorks, Inc.

Ertuğrul Keçeci and Tufan Kumbasar are with AI and Intelligent Systems Laboratory, Istanbul Technical University, 34469, Istanbul, Türkiye kececie@itu.edu.tr, kumbasart@itu.edu.tr

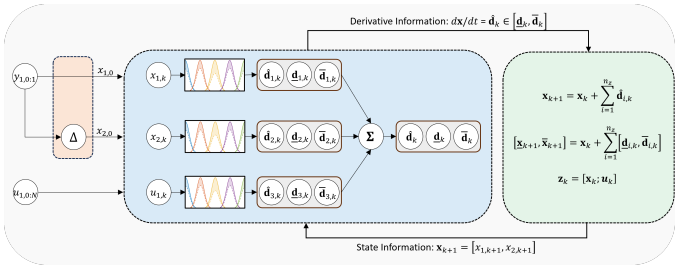


Fig. 1. Illustration of xFODE+ in a SISO setup with two states: Initial states are obtained from the State Representation block, defined as $\mathbf{x}_k = [y_k, \Delta y_k]$. Then, together with the input $u(k)$, they are fed to the IT2-FLSs to generate the derivative information, which is processed by the ODE block to compute the next states and the PI, i.e. $\mathbf{x}_{k+1} \in [\underline{\mathbf{x}}_{k+1}, \overline{\mathbf{x}}_{k+1}]$.

For reliable deployment of SysID models, UQ augments predictions with confidence information, commonly via Prediction Intervals (PIs) that bound unseen targets with confidence level δ [13]–[15]. FODE [8] incorporates UQ by modeling the antecedent space with Interval Type-2 (IT2) FSs, where each Membership Function (MF) is defined by a Footprint of Uncertainty (FOU) bounded by Lower and Upper MFs (LMF/UMF). Yet, in high-dimensional spaces, significant overlap among IT2-FSs can limit the interpretability of UQ.

This paper proposes *Explainable FODE+* (xFODE+), an interpretable data-driven SysID model that augments xFODE with UQ by producing PIs, as illustrated in Fig. 1. In xFODE+, states are represented in an incremental form to preserve the physically grounded temporal structure. To provide UQ, xFODE+ replaces Type-1 (T1) FLSs used in xFODE with IT2-FLSs. By aggregating Type-Reduced Sets (TRS) generated by each IT2-FLS, xFODE+ produces a state update together with its PI. For interpretability, we present PSs to the IT2-FLSs by constraining the MFs such that only two neighboring rules can fire at a time, limiting MF overlap and keeping inference locally transparent. We train xFODE+ end-to-end in a DL framework by minimizing a composite loss that combines predictive accuracy with PI quality. Comparisons with NODE, FODE, and xFODE on benchmark datasets show that xFODE+ achieves FODE-level PI quality while preserving NODE-level accuracy alongside interpretability.

II. Neural and Fuzzy ODE Networks

Consider a non-autonomous partially observable nonlinear system described by

$$\dot{\mathbf{x}}(t) = d(\mathbf{x}(t), \mathbf{u}(t)),\ \mathbf{x}(t_0) = \mathbf{x}_0,\ \mathbf{y}(t) = h(\mathbf{x}(t)) \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ denotes the state vector, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ the input vector, and $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ the outputs. $d(\cdot)$ defines the nonlinear dynamics, whereas $h(\cdot)$ maps states to outputs.

To model dynamics in (1), the evolution of states of NODEs and FODEs is governed by a parameterized vector field $d^{\mathrm{M}}(\cdot)$, where $\mathrm{M} \in \{\mathrm{NN}, \mathrm{T1}, \mathrm{IT2}\}$, defined by

$$\dot{\mathbf{x}}(t) = d^{\mathrm{M}}\left(\mathbf{x}(t), \mathbf{u}(t); \boldsymbol{\theta}\right) \tag{2}$$

where $\boldsymbol{\theta}$ denotes the Learnable Parameters (LPs). Given initial condition $\mathbf{x}(t_0)$ the state at t_1 is defined as

$$\mathbf{x}(t_1) = \mathbf{x}(t_0) + \int_{t_0}^{t_1} d^{\mathrm{M}}(\mathbf{x}(\tau),\ \mathbf{u}(\tau); \boldsymbol{\theta})\, d\tau. \tag{3}$$

Using Euler integration yields the discrete-time form of (1):

$$\mathbf{x}_{k+1} = \mathbf{x}_k + d^{\mathrm{M}}(\mathbf{z}_k; \boldsymbol{\theta}),\ \mathbf{y}_k = h(\mathbf{x}_k), \tag{4}$$

where $\mathbf{z}_k = [\mathbf{x}_k; \mathbf{u}_k] \in \mathbb{R}^{n_z}$ with $n_z = n_x + n_u$. In particular, for xFODE [12], the state update in (4) is in additive form:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \sum_{i=1}^{n_z} d_i^{\mathrm{T1}}(\mathbf{z}_{i,k}; \boldsymbol{\theta}_i) \tag{5}$$

When FODE is defined with IT2-FLSs, it outputs the TRS $\left[\underline{\mathbf{d}}_k, \overline{\mathbf{d}}_k\right]$ alongside crisp output [8]:

$$\left[\hat{\mathbf{d}}_k, \underline{\mathbf{d}}_k, \overline{\mathbf{d}}_k\right] = d^{\mathrm{IT2}}(\mathbf{z}_k; \boldsymbol{\theta}), \tag{6}$$

Accordingly, IT2-FODE yields a PI for the next state:

$$\left[\underline{\mathbf{x}}_{k+1}, \overline{\mathbf{x}}_{k+1}\right] = \mathbf{x}_k + \left[\underline{\mathbf{d}}_k,\ \overline{\mathbf{d}}_k\right], \tag{7}$$

while the state update in (4) uses $\hat{\mathbf{d}}_k = \left(\underline{\mathbf{d}}_k + \overline{\mathbf{d}}_k\right)/2$. We refer the reader to [1], [8], [12] for more details.

III. xFODE+ Framework

This section introduces xFODE+, which augments xFODE with UQ capability via PIs (Fig. 1).

A. Interpretable State Representation

A standard approach for constructing the state vector $\mathbf{x}_k$ of NODE/FODE is directly from measured outputs [8]:

SR0: $$\mathbf{x}_k = \mathbf{y}_k, \tag{8}$$

which is an interpretable state, as $\mathbf{x}_k$ corresponds to physical output variables. Yet, it does not encode temporal dependencies, which are essential for capturing complex dynamics.

To encode temporal dependencies while preserving interpretability, we suggest defining $\mathbf{x}_k$ in the *incremental form*:

SR1: $$\mathbf{x}_k = [\,\mathbf{y_k}, \Delta\mathbf{y}_k, \ldots, \Delta^m\mathbf{y}_k\,]^T \tag{9}$$

with m denoting the number of difference orders and $\Delta^j\mathbf{y}_k = \Delta^{j-1}\mathbf{y}_k - \Delta^{j-1}\mathbf{y}_{k-1}, j = 1, \ldots, m$. This representation emphasizes temporal changes, with each term corresponding to a physically meaningful quantity, such as velocity or acceleration of the outputs.

B. Additive ODE Dynamics

The state derivative in xFODE+ is represented additively.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \sum_{i=1}^{n_z} \hat{\mathbf{d}}_{i,k}, \tag{10}$$

where $\hat{\mathbf{d}}_{i,k} \in \mathbb{R}^{n_x}$ represents the contribution of $z_{i,k}$. Each contribution is produced by $f_i(\cdot)$, which outputs both a point prediction and a corresponding TRS:

$$\left[\hat{\mathbf{d}}_{i,k}, \underline{\mathbf{d}}_{i,k}, \overline{\mathbf{d}}_{i,k}\right] = f_i(z_{i,k}, \boldsymbol{\theta}_i). \tag{11}$$

To obtain a PI for the next state, we define:

$$\left[\underline{\mathbf{x}}_{k+1}, \overline{\mathbf{x}}_{k+1}\right] = \mathbf{x}_k + \left[\sum_{i=1}^{n_z} \underline{\mathbf{d}}_{i,k},\ \sum_{i=1}^{n_z} \overline{\mathbf{d}}_{i,k}\right]. \tag{12}$$

This additive structure allows xFODE+ to provide input-wise interpretability while generating PIs.

C. Interpretable Mapping Design

In xFODE+, each $f_i(.)$ in (11) is implemented as a single-input IT2-FLS that maps $f_i : \mathbb{R} \to \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$. For notational simplicity, we drop the indices i and k.

The IT2-FLS has P rules ($p = 1, \ldots, P$), defined as

$$R_p:\ \text{If } z \text{ is } \hat{A}_p \text{ then } \hat{\mathbf{d}} \text{ is } \mathbf{d}_p(z), \tag{13}$$

where $\mathbf{d}_p(z) = [d_{p,1}(z), \ldots, d_{p,n_x}(z)]^T$ and $d_{p,o}(z) = a_p^o z + a_{p,0}^o$, $o = 1, \ldots, n_x$. The crisp output is calculated as $\hat{\mathbf{d}} = \left(\underline{\mathbf{d}} + \overline{\mathbf{d}}\right)/2$, where $\underline{\mathbf{d}} = [\underline{d}^{(1)}, \ldots, \underline{d}^{(n_x)}]^T$ and $\overline{\mathbf{d}} = [\overline{d}^{(1)}, \ldots, \overline{d}^{(n_x)}]^T$. The TRS is obtained via the KM algorithm, which is performed for each output separately.

$$\begin{aligned}
\underline{d}^{(o)} &= \frac{\sum_{p=1}^{L} \overline{\mu}_p(z)\, d_{p,o}(z) + \sum_{p=L+1}^{P} \underline{\mu}_p(z)\, d_{p,o}(z)}{\sum_{p=1}^{L} \overline{\mu}_p(z) + \sum_{p=L+1}^{P} \underline{\mu}_p(z)} \\
\overline{d}^{(o)} &= \frac{\sum_{p=1}^{R} \underline{\mu}_p(z)\, d_{p,o}(z) + \sum_{p=R+1}^{P} \overline{\mu}_p(z)\, d_{p,o}(z)}{\sum_{p=1}^{R} \underline{\mu}_p(z) + \sum_{p=R+1}^{P} \overline{\mu}_p(z)}
\end{aligned} \tag{14}$$

Here, L and R are the switching points of the KM algorithm.

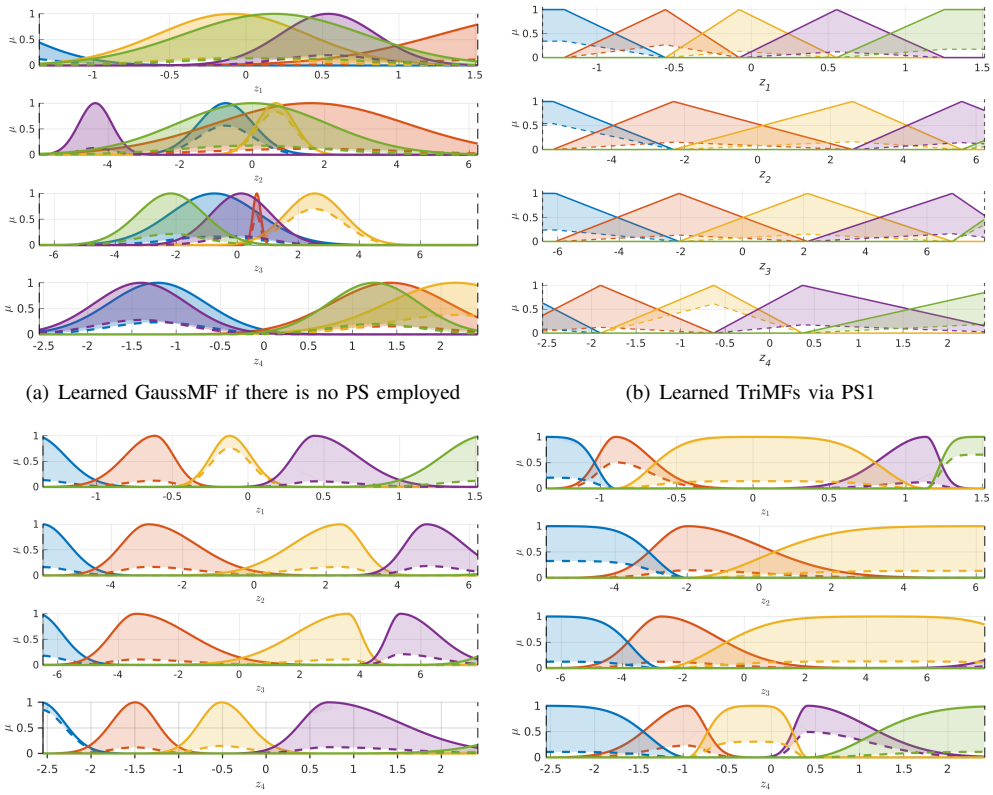
In (14), $\underline{\mu}_p(z)$ and $\overline{\mu}_p(z)$ are the membership grades of the LMF and UMF. Gaussian MFs (GaussMFs) are commonly used due to their capacity to model nonlinearities.

$$\overline{\mu}_p(z) = \exp(-(z - c_p)^2/2(\sigma_p)^2),\ \underline{\mu}_p(z) = h_p\overline{\mu}_p(z) \tag{15}$$

where c_p and σ_p denote the center and standard deviation, respectively, and $h_p \in (0, 1]$ defines the height of the LMF and thus the size of the FOU. As shown in Fig. 2(a), if there are no PSs, their overlap can cause multiple rules to be simultaneously fired, reducing interpretability.

To address this, we propose PSs to sculpt the antecedent space so that only two consecutive rules are activated. As a result, for $z' \in [c_{p^*}, c_{p^*+1}]$, the KM is performed with $N^* = 2$, yielding $L = R = 1$ [16], which simplifies (14) to:

$$\begin{aligned}
\underline{d}^{(o)} &= \frac{\overline{\mu}_{p^*}(z')\, d_{p^*,o}(z') + \underline{\mu}_{p^*+1}(z')\, d_{p^*+1,o}(z')}{\overline{\mu}_{p^*}(z') + \underline{\mu}_{p^*+1}(z')}, \\
\overline{d}^{(o)} &= \frac{\underline{\mu}_{p^*}(z')\, d_{p^*,o}(z') + \overline{\mu}_{p^*+1}(z')\, d_{p^*+1,o}(z')}{\underline{\mu}_{p^*}(z') + \overline{\mu}_{p^*+1}(z')}.
\end{aligned} \tag{16}$$



(a) Learned GaussMF if there is no PS employed

(b) Learned TriMFs via PS1

(c) Learned Gauss2MF via PS2

(d) Learned Gauss2MF via PS3

Fig. 2. Visualization of learned IT2-FSs on the MR Damper dataset (single seed). States are defined with the incremental form (SR1), thus $\mathbf{z} = [\mathbf{y}; \Delta\mathbf{y}; \Delta^2\mathbf{y}; \mathbf{u}]$. Each combined input dimension z_i is interpretable: z_1 output position, z_2 output velocity, z_3 output acceleration, and z_4 control input.

This ensures a simple mapping while maintaining interpretability. For all presented PSs, we define LMFs as in (15).

1) PS1-Partitioning with Triangular MFs (TriMFs): TriMFs enable a simple partitioning; the UMFs are set as

$$\overline{\mu}_p(z) = \max\left(0, \min\left(\frac{z - l_p}{c_p - l_p}, \frac{r_p - z}{r_p - c_p}\right)\right) \tag{17}$$

where l_p, c_p, and r_p denote the left feet, center, and right feet. For partitioning, the following couplings are defined:

$$\begin{aligned} \Delta_p^l = c_p - l_p, \quad \Delta_p^r = r_p - c_p, \\ c_{p+1} = c_p + \Delta_p^r, \quad \Delta_{p+1}^l = \Delta_p^r. \end{aligned} \tag{18}$$

This coupling ensures that, for any input z', only two consecutive UMFs are activated as shown in Fig. 2(b).

2) PS2-Partitioning with Two-Sided Gaussian MFs (Gauss2MFs): To provide flexibility and smoothness, we employ two-sided Gaussian MFs (Gauss2MFs) as UMFs:

$$\overline{\mu}_p(z) = \begin{cases} \exp\left(-\left(z - c_p\right)^2 / 2\left(\sigma_p^l\right)^2\right), & \text{if } z \le c_p \\ \exp\left(-\left(z - c_p\right)^2 / 2\left(\sigma_p^r\right)^2\right), & \text{if } z > c_p \end{cases} \tag{19}$$

where σ_p^l and σ_p^r are the left and right standard deviations, respectively. For interpretability, we parameterize it as:

$$c_{p+1} = c_p + 4\sigma_p^r, \quad \sigma_p^r = \sigma_{p+1}^l \tag{20}$$

This results in a partition as shown in Fig. 2(c). Note that we assume $\overline{\mu}_p(z') \approx 0$ when $|z'| > 4\sigma_p^r$ [11].

3) PS3-Partitioning with Complementary Gauss2MFs: In PS2, while Gauss2MFs offer asymmetric flexibility, the 4σ center spacing in (20) may result in weak overlap between adjacent UMFs. Thus, to increase the overlap, we construct odd-indexed UMFs as local complements of neighboring even-indexed UMFs. Let even indexed A_{2q} ($q = 1, 2, \ldots, \lceil (P-3)/2 \rceil$) have UMFs $\overline{\mu}_{2q}(z)$ defined by (19). The midpoint between consecutive even-indexed centers is:

$$c_{2q+1}^{\text{mid}} = (c_{2q} + c_{2(q+1)})/2. \tag{21}$$

The odd indexed UMFs $\overline{\mu}_{2q+1}(z)$ are then defined as

$$\overline{\mu}_{2q+1}(z) = \begin{cases} 0, & z < c_{2q}, \\ 1 - \overline{\mu}_{2q}(z), & c_{2q} \le z \le c_{2q+1}^{\text{mid}}, \\ 1 - \overline{\mu}_{2(q+1)}(z), & c_{2q+1}^{\text{mid}} \le z \le c_{2(q+1)}, \\ 0, & z > c_{2(q+1)}, \end{cases} \tag{22}$$

with $\overline{\mu}_1(z) = 1 - \overline{\mu}_2(z), z < c_2$ and when P is odd, $\overline{\mu}_P(z) = 1 - \overline{\mu}_{P-1}(z), z > c_{P-1}$. This UMF design guarantees that only two UMFs are active as shown in Fig. 2(d).

IV. DL Framework for xFODE+

We next introduce a DL framework to train xFODE+. Algorithm 1* outlines the training procedure and Algorithm

*MATLAB implementation. [Online]. Available: https://github.com/ertugrulkececi/xfodeplus

Algorithm 1 Training steps of xFODE+

Input: Dataset $D = \{\mathbf{u}_k, \mathbf{y}_k\}_{k=1}^{K}$, prediction horizon N, number of rules P, mini-batch size mbs, number of epochs E, number of difference orders m, desired quantile levels $\{\underline{\tau}, \overline{\tau}\}$
Output: LP set θ
Initialize θ;
Construct $\mathbf{x}_k$ via (9);
Build $S = (\mathbf{u}_{1:N}^{\{j\}}, \mathbf{x}_{1:N}^{\{j\}})_{j=1}^{B}$;
for $e = 1$ to E **do**
  for each mbs in B **do**
    Select a mini-batch $[\mathbf{u}_{0:N}^{\{j\}}, \mathbf{x}_{0:N}^{\{j\}}]_{j=1}^{mbs}$
    $\left[\hat{\mathbf{x}}_{1:N}^{\{j\}}, \underline{\mathbf{x}}_{1:N}^{\{j\}}, \overline{\mathbf{x}}_{1:N}^{\{j\}}\right] \leftarrow$ xFODE+$([\mathbf{x}_0^{\{j\}}, \mathbf{u}_{0:N}^{\{j\}}]; \theta)$
    Compute L_C and the gradient $\partial L_C / \partial \theta$
    Update θ via a DL optimizer, e.g., Adam
  end for
end for
$\theta^* = \arg\min(L_C)$
Return $\theta = \theta^*$

Algorithm 2 Inference of xFODE+ for a trajectory

Input: Initial state $\mathbf{x_0}$, input sequence $\mathbf{u}_{0:N}$, LP set θ, prediction horizon N
Output: Predicted state trajectory $\hat{\mathbf{x}}_{1:N}$
Initialize $\left[\hat{\mathbf{x}}, \underline{\mathbf{x}}, \overline{\mathbf{x}}\right] \leftarrow \varnothing$, $\hat{\mathbf{x}}_0 = \mathbf{x}_0$
for $k = 0$ **to** $N - 1$ **do**
  $\mathbf{z_k} \leftarrow [\hat{\mathbf{x}}_k, \mathbf{u}_k]$
  $\left[\hat{\mathbf{d}}_k, \underline{\mathbf{d}}_k, \overline{\mathbf{d}}_k\right] \leftarrow \sum_{i=1}^{n_z} f_i(z_{i,k}, \boldsymbol{\theta}_i)$
  $\left[\hat{\mathbf{x}}_{k+1}, \underline{\mathbf{x}}_{k+1}, \overline{\mathbf{x}}_{k+1}\right] \leftarrow \hat{\mathbf{x}}_k + \left[\hat{\mathbf{d}}_k, \underline{\mathbf{d}}_k, \overline{\mathbf{d}}_k\right]$
  $\left[\hat{\mathbf{x}}, \underline{\mathbf{x}}, \overline{\mathbf{x}}\right] \leftarrow \left[\hat{\mathbf{x}}, \underline{\mathbf{x}}, \overline{\mathbf{x}}\right] \cup \left[\hat{\mathbf{x}}_{k+1}, \underline{\mathbf{x}}_{k+1}, \overline{\mathbf{x}}_{k+1}\right]$
  $\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_{k+1}$
end for
Return $\left[\hat{\mathbf{x}}, \underline{\mathbf{x}}, \overline{\mathbf{x}}\right]$

2 describes the simulation inference. We consider a dataset $D = \{\mathbf{u}_k, \mathbf{y}_k\}_{k=1}^{K}$. After defining $\mathbf{x}_k$, the resulting input–state trajectories for $j \in [1, \ldots, B$ are:

$$S = \left[\left(\mathbf{u}_0^{\{j\}}, \mathbf{x}_0^{\{j\}}\right), \ldots, \left(\mathbf{u}_N^{\{j\}}, \mathbf{x}_N^{\{j\}}\right)\right] \tag{23}$$

Here, N is the roll-out while B is the number of trajectories.

A. Loss Function

xFODE+ learns the underlying system dynamics by minimizing a composite loss function over S:

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\mathcal{C}}} L_C = L_A + L_{UQ} \tag{24}$$

where the accuracy loss

$$L_A = \frac{1}{B} \sum_{j=1}^{B} \sum_{k=1}^{N} |\mathbf{x}_k^{\{j\}} - \hat{\mathbf{x}}_k^{\{j\}}| \tag{25}$$

To train xFODE+ to produce valid PIs at each time step, the TRS $C_k = [\underline{\mathbf{x}}_k, \overline{\mathbf{x}}_k]$. are required to satisfy the marginal coverage condition $\mathbb{P}\{\mathbf{x}_k \in C_k\} \geq \delta$. To learn such PIs, we employ the tilted (pinball) loss. For a residual e and quantile level $\tau \in (0, 1)$, the tilted loss is defined as $\rho_\tau(e) = \max(\tau e, (\tau - 1)e)$. The resulting UQ loss is defined as

$$L_{UQ} = \frac{1}{B} \sum_{j=1}^{B} \sum_{k=1}^{N} \left(\rho_{\underline{\tau}}(\underline{e}_k^{(j)}) + \rho_{\overline{\tau}}(\overline{e}_k^{(j)})\right), \tag{26}$$

with $\underline{e}_k^{\{j\}} = \mathbf{x}_k^{\{j\}} - \underline{\mathbf{x}}_k^{\{j\}}$ and $\overline{e}_k^{\{j\}} = \mathbf{x}_k^{\{j\}} - \overline{\mathbf{x}}_k^{\{j\}}$. Here, $\underline{\tau}$ and $\overline{\tau}$ denote the lower and upper quantile levels. For a given δ, we set $\underline{\tau} = (1 - \delta)/2$ and $\overline{\tau} = 1 - (1 - \delta)/2$.

B. Parameterization Tricks

The complete set of LPs is defined as $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}^{n_z}$ where each $\boldsymbol{\theta}_i$ is parameterized according PS:

- For PS1: $\boldsymbol{\theta}_i = \{c_1, \Delta_1^l, \boldsymbol{\Delta}^r, \mathbf{h}, \mathbf{d}\}$ where c_1 and Δ_1^l are scalar, $\boldsymbol{\Delta}^r \in \mathbb{R}^{P \times 1}$, $\mathbf{h} \in \mathbb{R}^{P \times 1}$, and $\mathbf{d} \in \mathbb{R}^{2Pn_x \times 1}$,
- For PS2/ PS3: $\boldsymbol{\theta}_i = \{c_1, \sigma_1^l, \boldsymbol{\sigma}^r, \mathbf{h}, \mathbf{d}\}$ where c_1 and σ_1^l σ are scalar, $\boldsymbol{\sigma}^r \in \mathbb{R}^{P \times 1}$, $\mathbf{h} \in \mathbb{R}^{P \times 1}$, and $\mathbf{d} \in \mathbb{R}^{2Pn_x \times 1}$

During training, it is necessary to enforce $\sigma_p > 0$ or $\Delta_p > 0$, $\forall p$ [17]. This is achieved via softplus reparameterization, i.e., $\Delta_p = \log(1 + e^{\Delta_p'})$ for TriMFs and $\sigma_p = \log(1 + e^{\sigma_p'})$ for Gauss2MFs, while h_p is constrained to $h_p \in (0.1, 1)$ using a scaled sigmoid mapping $h_p = 0.1 + 0.9(1 + e^{-h_p'})^{-1}$ to limit the FOU size and improve computational stability.

V. Performance Analysis

We evaluate the SysID performance of xFODE+ on three benchmark datasets: Hair Dryer, MR Damper, and Steam Engine. All datasets are normalized prior to training and split into training/test as in [12]. Experiments are conducted in MATLAB and repeated over 20 independent runs. All models are evaluated in simulation (multi-step prediction) mode with a roll-out length of $N = 20$. For UQ, a target coverage of 99% is defined. Model accuracy is quantified using RMSE, while the quality of PIs is evaluated using the PI Coverage Percentage (PICP) and the PI Normalized Average Width (PINAW) [14].

The following models are considered:

- **NODE:** $d^{\mathrm{NN}}(\cdot)$ is a neural network with two hidden layers of 128 units each and *tanh* activations. The NODE models are trained with L_A as training loss.
- **T1-FODE:** $d^{\mathrm{T1}}(\cdot)$ is a multi-input T1-FLS using GaussMFs. The models are trained with L_A [8].
- **xFODE:** The models are learned with PS1–PS3 and L_A as training loss [12].
- **IT2-FODE:** $d^{\mathrm{IT2}}(\cdot)$ is a multi-input IT2-FLS, using GaussMFs. The models are trained with L_C [8].
- **xFODE+:** Each IT2-FLS $f_i(\cdot)$ in (11) is constructed using PS1–PS3 and trained as described in Section IV.
- **Additive FODE+:** To assess the effect of PSs, we train xFODE+ using GaussMFs without any PS (AFODE+).

We trained all models with both the SR0 and SR1 representations. The hyperparameter of SR1 m is selected via cross-validation on NODE and fixed across all models for fairness. The resulting values are $m = 2$ for the Hair Dryer and MR Damper datasets, and $m = 1$ for the Steam Engine dataset. All fuzzy models are defined with $P = 5$ rules.

We observe that models trained with the SR0 consistently perform poorly across all datasets, as illustrated for the Hair Dryer dataset in Fig. 3. Thus, we focus exclusively on SR1 models. Table I reports the corresponding mean RMSE, uncertainty metrics, and #LP. We conclude that:

- xFODE+ generates PIs comparable to IT2-FODE on MR Damper and Steam Engine across all PSs (Figs. 4–5), with slightly wider PIs, while achieving similar RMSEs.
- AFODE+ yields PIs that achieve target coverage, while obtaining low RMSE. However, its learned MFs may overlap (Fig. 2(a)), whereas xFODE+ restricts activation to two neighboring MFs (Figs. 2(b)–2(d)).
- NODE and T1-FODE achieve slightly lower RMSE, but they are optimized for accuracy only. xFODE+ instead optimizes a composite loss that also accounts for PI quality with far fewer #LP than NODE.
- xFODE shows superior SysID performance than xFODE+, indicating a trade-off between predictive accuracy and UQ capability introduced in xFODE+.
- On the MR Damper dataset, xFODE+ yields similar PIs across all PSs. On the Hair Dryer and Steam Engine datasets, PS1 gives PIs closest to the target coverage compared to PS2 or PS3.

Overall, xFODE+ enhances interpretability through structured PSs while maintaining PI quality comparable to IT2-FODE, with similar RMSE and #LP.

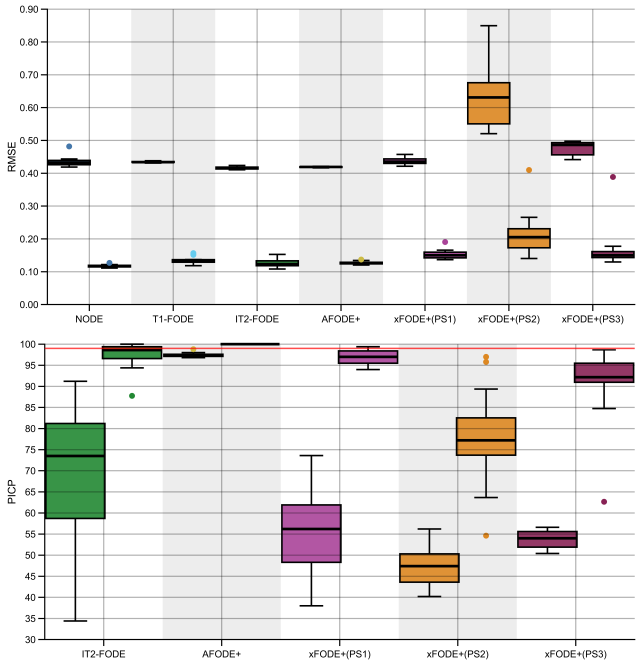


Fig. 3. RMSE and PICP boxplots of trained models on Hair Dryer dataset. For each model, the left boxplot corresponds to SR0, and the right boxplot corresponds to SR1. The horizontal red line depicts the desired coverage.

VI. Conclusion and Future Work

This paper introduced xFODE+, an explainable SysID framework that extends xFODE with UQ by leveraging IT2-FLSs to generate PIs. It represents states in incremental form to preserve physical meaning, and aggregates the TRSs from each IT2-FLS to compute the state update and its PI. Using structured antecedent partitioning, xFODE+ yields transparent IT2-FSs with clean linguistic semantics (e.g., Negative–Zero–Positive) (Fig. 2). While xFODE+ offers such an enhancement, it maintains accuracy and UQ performance comparable to IT2-FODE, with a similar #LP.

Future work will focus on analyzing the interpretability of the rule consequents and extending the evaluation of xFODE+ to a broader range of datasets.

Acknowledgment

The authors acknowledge using ChatGPT to refine the grammar and enhance the expression of English.

References

[1] G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. B. Schön, "Deep networks for system identification: a survey," *Automatica*, vol. 171, p. 111907, 2025.

[2] T. Dai, K. Aljanaideh, R. Chen, R. Singh, A. Stothert, and L. Ljung, "Deep learning of dynamic systems using system identification toolbox™," *IFAC-PapersOnLine*, vol. 58, no. 15, pp. 580–585, 2024.

[3] M. Forgione, A. Muni, D. Piga, and M. Gallieri, "On the adaptation of recurrent neural networks for system identification," *Automatica*, vol. 155, p. 111092, 2023.

[4] A. Rahman, J. Drgoňa, A. Tuor, and J. Strube, "Neural ordinary differential equations for nonlinear system identification," in *IEEE American control conference*, 2022, pp. 3979–3984.

[5] N. Gashi, P. Kakosimos, and G. Papafotiou, "System identification using kolmogorov-arnold networks: A case study on buck converters," in *IEEE International Conference on Prognostics and Health Management*, 2025.

[6] Y. Yang and H. Li, "Neural ordinary differential equations for robust parameter estimation in dynamic systems with physical priors," *Applied Soft Computing*, vol. 169, p. 112649, 2025.

[7] L. Böttcher, "Gradient-free training of neural odes for system identification and control using ensemble kalman inversion," *arXiv preprint arXiv:2307.07882*, 2023.

[8] Y. Güven and T. Kumbasar, "Fuzzy logic strikes back: Fuzzy odes for dynamic modeling and uncertainty quantification," *IEEE Transactions on Artificial Intelligence*, 2025.

[9] E. Mariotti, J. M. A. Moral, and A. Gatt, "Exploring the balance between interpretability and performance with carefully designed constrainable neural additive models," *Information Fusion*, vol. 99, p. 101882, 2023.

[10] Z. Yang, A. Zhang, and A. Sudjianto, "Gami-net: An explainable neural network based on generalized additive models with structured interactions," *Pattern Recognition*, vol. 120, p. 108192, 2021.

[11] Ö. B. Gökmen, Y. Güven, and T. Kumbasar, "FAME: Introducing fuzzy additive models for explainable ai," in *IEEE International Conference on Fuzzy Systems*, 2025.

[12] E. Keçeci and T. Kumbasar, "xFODE: An explainable fuzzy additive ode framework for system identification," in *IEEE Conference on Artificial Intelligence*, 2026, accepted.

[13] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic learning in a random world*. Springer, 2005.

[14] S. Msaddi and T. Kumbasar, "An alliance for uncertainty quantification: Type-2 fuzzy logic systems with conformal prediction," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2026.

[15] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, "A survey of uncertainty in deep neural networks," *Artificial Intelligence Review*, vol. 56, pp. 1513–1589, 2023.

[16] T. Kumbasar, "Robust stability analysis and systematic design of single-input interval type-2 fuzzy logic controllers," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 3, pp. 675–694, 2015.

[17] A. Beke and T. Kumbasar, "More than accuracy: A composite learning framework for interval type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 31, no. 3, pp. 734–744, 2022.

TABLE I
MEAN TESTING PERFORMANCE METRICS OVER 20 EXPERIMENTS

Model	**Hair Dryer** ($n_u = 1$, $n_y = 1$)				**MR Damper** ($n_u = 1$, $n_y = 1$)				**Steam Engine** ($n_u = 2$, $n_y = 2$)		
	#LP	RMSE	PICP	PINAW	#LP	RMSE	PICP	PINAW	#LP	RMSE	PICP / PINAW
NODE	17539	0.1173	–	–	17539	9.6809	–	–	17924	y_1: 0.0909 y_2: 0.0873	y_1: – / – y_2: – / –
FODE	115	0.1352	–	–	115	9.2819	–	–	200	y_1: 0.1014 y_2: 0.1129	y_1: – / – y_2: – / –
xFODE(PS1)	148	0.1271	–	–	148	9.6841	–	–	282	y_1: 0.0790 y_2: 0.0729	y_1: – / – y_2: – / –
xFODE(PS2)	148	0.1249	–	–	148	9.9384	–	–	282	y_1: 0.0802 y_2: 0.0740	y_1: – / – y_2: – / –
xFODE(PS3)	148	0.1293	–	–	148	10.0630	–	–	282	y_1: 0.0862 y_2: 0.0768	y_1: – / – y_2: – / –
IT2-FODE	135	0.1258	97.60	0.270	135	10.1466*	96.56	0.417	230	y_1: 0.0864 y_2: 0.0979	y_1: 96.72 / 0.247 y_2: 94.10 / 0.683
AFODE+	180	0.1268	100.00	0.431	180	10.2086	98.81	0.411	330	y_1: 0.0785 y_2: 0.0770	y_1: 99.85 / 0.242 y_2: 99.35 / 0.732
xFODE+(PS1)	168	0.1524	96.84	0.310	168	13.5024	95.24	0.418	312	y_1: 0.0917 y_2: 0.0879	y_1: 98.55 / 0.276 y_2: 96.27 / 0.679
xFODE+(PS2)	168	0.2092	77.92	0.267	168	11.6543	96.07	0.452	312	y_1: 0.1100 y_2: 0.1105	y_1: 95.40 / 0.270 y_2: 86.97 / 0.591
xFODE+(PS3)	168	0.1625	91.41	0.317	168	13.2810	96.59	0.489	312	y_1: 0.1096 y_2: 0.1034	y_1: 97.55 / 0.308 y_2: 91.92 / 0.689

* IT2-FODE yielded NaN values for 2 seeds. These experiments are excluded from statistical analysis.

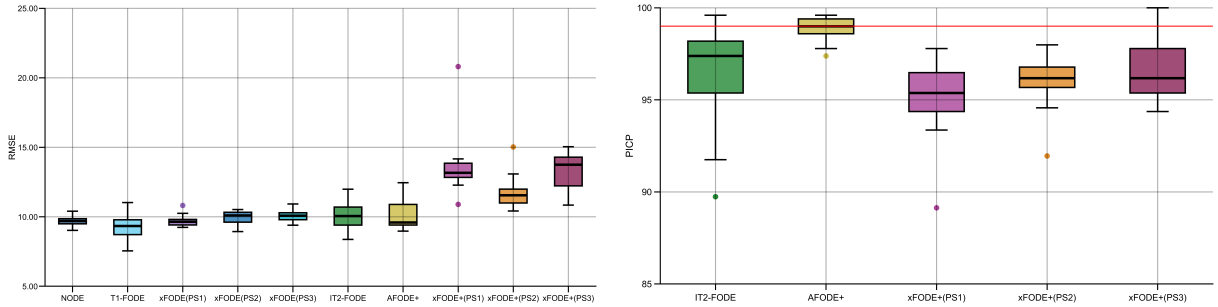


Fig. 4. RMSE and PICP boxplots of trained models on the MR Damper dataset. The horizontal red line depicts the desired coverage.

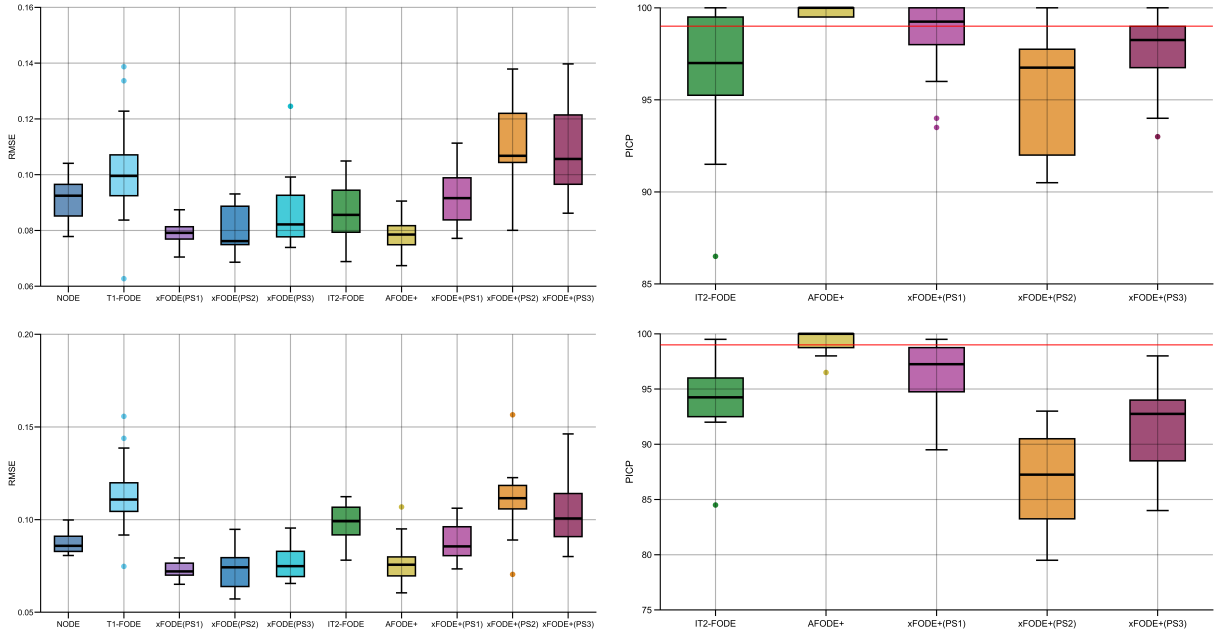


Fig. 5. RMSE and PICP boxplots of trained models on the Steam Engine dataset; top row corresponds to y_1 and bottom row corresponds to y_2. The horizontal red line depicts the desired coverage.