STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–13.
Published online in International Academic Press (www.IAPress.org)

Enhancing fuzzy transform using PCHIP interpolation: A novel approach to function approximation and solving differential equations

Ashwaq Abdul Qadir Khidr [1], Edrees M. Nori Mahmood [2,*]

[1] *Department of Mathematics, College of Computer Science and Mathematics, University of Mosul, Iraq*
[2] *Department of Operations Research and Intelligent Techniques, College of Computer Science and Mathematics, University of Mosul, Iraq*

Abstract In this study, a hybrid numerical method is presented, combining Fuzzy Transform (FT) and PCHIP (Piecewise Cubic Hermite Interpolating Polynomial) interpolation techniques in developing the accuracy and flexibility of function approximation and solutions to differential equations. The method operates in two stages: first, a low-dimensional fuzzy approximation is constructed using basis functions on a coarse grid, capturing global trends efficiently. Second, residuals between the fuzzy approximation and the true solution (or observed data) are interpolated using PCHIP, which preserves monotonicity and local shape characteristics while avoiding spurious oscillations. Numerical validation demonstrates a reduction of over 98% in mean square error compared to the standalone fuzzy transform, confirming the enhanced accuracy of the improved method across the tested cases. Theoretical error bounds are derived via the superposition principle, demonstrating that the total error is governed by the sum of FT approximation and PCHIP interpolation errors. Using this method, discrete measurements or sample observations can be mathematically modeled, and the method creates an interpolant in continuous space for any empirical data by using PCHIP, which makes it possible for any real-world data sets to be treated analytically (e.g., differentiated or integrated) over the observed values. So, this spoken itself satisfies the gap between measurements taken as discrete observations and using continuous representations in modeling. This would be highly useful for experimental science and engineering applications, such as when retrieving sensor data or performing irregularly sampled measurements that need intensive numerical treatment.

Keywords Fuzzy Transform, PCHIP Interpolation, Error Correction, Function Approximation, Differential Equations

DOI: 10.19139/soic-2310-5070-2624

1. Introduction

The FT, a method developed by Perfilieva [9, 10], is an attempt to merge Zadeh's theoretical framework of fuzzy set theory [21] with functional analysis for function approximation and efficient data processing. The methods based on classical orthogonal basis, such as Fourier or Wavelet Transforms, independently work at capturing local data variations using fuzzy partitions and balance them with the whole data sample [1, 9, 13]. Because of such an ability, the FT has gained tremendous importance in handling uncertainty for many applications like signal/image processing [3, 4, 5, 11], data analysis [2, 15, 16], and noise reduction. But then, some of the classical FT methods suffer from limitations: (i) Sensitivity against partition design; improper node placement or too little overlap would compromise stability of approximation [1, 10]; (ii) difficulties at the edges produce boundary discontinuities—this occurs if one does not handle edge regions properly with special techniques like reflection or extended partitions [10, 12]; (iii) ever-decreasing efficiency under nonlinear or high-dimensional

*Correspondence to: Edrees M. Nori Mahmood (Email: edreesnori@uomosul.edu.iq). Department of Operations Research and Intelligent Techniques, College of Computer Science and Mathematics, University of Mosul. Iraq (000000).

ISSN 2310-5070 (online) ISSN 2311-004X (print)

setups with increasing computational complexity—such that approximation would give in for lack of support from hybrid frameworks or higher degree extensions [8, 14, 20]. The presented work develops a promising two-phase hybrid technique, linking FT with the PCHIP to overcome the mentioned disadvantages. In its first phase, a low-order FT approximation is created over a coarse grid due to its efficiency in capturing global trends [9, 14]. The residuals between the FT approximation and the real solution are then interpolated through PCHIP [6, 7]; thereby, monotonicity and characteristics of local shape are preserved, while spurious oscillations that often plague classical interpolation methods are evaded [18]. Key insights include the exclusion of fuzzy nodes in residual interpolation to prevent overfitting in accordance with residual-driven adaptation in FT [17], and to exert initial/boundary conditions explicitly to ensure physics consistency [8]. Numerical validation shows that compared to independent FT, the current method yields a 98% mean square error reduction while outperforming the existing hybrid approaches [8, 19, 20]. This progress also connects theoretical fuzzy methodologies [21, 22] with computability; thus, this presents a powerful framework to tackle complex data-driven problems. The organization of this paper proceeds as follows: Section 2 examines the theoretical and mathematical groundwork underpinning the proposed framework, establishing essential definitions and principles necessary for deriving the contributions outlined in later sections. Section 3 introduces a novel approach to enhancing the FT methodology through PCHIP-driven residual correction. This technique is systematically derived, with its algorithmic workflow and theoretical basis rigorously formalized. Section 4 evaluates the proposed methodology via comprehensive numerical simulations, benchmark validations, and comparative analyses to verify its computational efficacy and adaptability across diverse scenarios. Section 5 synthesizes the core findings, discusses their significance for practical applications, and identifies prospective avenues for advancing research in this domain.

2. Basic concepts

Before setting into all the technical minutiae of the enhancements proposed, it is important to review the basic principles of FT. This section offers the required mathematics background and sets the notation that will be used throughout the paper. Interested readers are redirected with the rest of the information given through [10, 12, 21, 22].

2.1. Fuzzy sets and membership functions

A fuzzy set A is defined through its membership function with the following properties: $\mu_A \colon X \to [0,1]$ represents the degree of membership of an element x in A. Different types of membership functions—used by fuzzy logic applications—include triangular, trapezoidal, Gaussian, and bell-shape forms. The triangular membership function is expressed as follows:

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \frac{x-a_1}{a_2-a_1}, & a_1 \le x \le a_2 \\ \frac{a_3-x}{a_3-a_2}, & a_2 \le x \le a_3 \\ 0, & x > a_3 \end{cases}$$

the parameters a_1, a_2 , and a_3 set the shape of the function.

2.2. Fuzzy partitions

On a closed real interval $[a,b]$, consider the fixed nodes $x_1, x_2, \ldots, x_n$, such that $x_1 = a$, $x_n = b$, and $n \ge 2$. The fuzzy sets $A_1, A_2, \ldots, A_k$, identified with $\mu_{A_1}(x), \mu_{A_2}(x), \ldots, \mu_{A_n}(x)$ on $[a,b]$, construct a fuzzy partition in $[a,b]$ if, for every $k = 1, 2, \ldots, n$, the following holds:

1. $\mu_{A_k}(x) : [a,b] \to [0,1]$ is continuous and $\mu_{A_k}(x_k) = 1$.
2. $\mu_{A_k}(x) = 0$ if $x \notin (x_{k-1}, x_{k+1})$ where $x_{k-1} = a$, $x_{k+1} = b$.
3. In $[x_{k-1}, x_k]$, the function $\mu_{A_k}(x)$ monotonically increases, while in $[x_k, x_{k+1}]$, it monotonically decreases.

4. $\sum_{k=1}^{n} \mu_{A_k}(x) = 1$ for all $x \in [a, b]$.

Membership functions $\mu_{A_1}(x), \mu_{A_2}(x), \ldots, \mu_{A_n}(x)$ referred to as basic functions. A fuzzified partition is defined as uniform when its nodes $x_1, x_2, \ldots, x_n$ exhibit equidistant spacing. Mathematically, the position of the k-th node is expressed as

$$x_k = a + h(k-1),$$

where h denotes the fixed distance between adjacent nodes, computed as

$$h = \frac{b-a}{n-1}.$$

This framework holds for integers $n \geq 2$ and indices $k = 1, 2, \ldots, n$. For the partition to qualify as uniform, two essential properties must also be satisfied:

5. **Symmetry:** The membership function $\mu_{A_k}(x)$ is symmetric about its respective node x_k, satisfying the condition
$$\mu_{A_k}(x_k - x) = \mu_{A_k}(x_k + x) \quad \text{for all } x.$$
This applies to nodes x_k, $k = 2, \ldots, n-1$ when $n > 2$.
6. **Shift Invariance:** The membership function of the subsequent node $\mu_{A_{k+1}}(x)$ corresponds to a shifted version of $\mu_{A_k}(x)$ by the distance h, such that
$$\mu_{A_{k+1}}(x) = \mu_{A_k}(x - h).$$
This property holds for all x and indices $k = 2, \ldots, n-2$ when $n > 2$.

Figure 1 visually demonstrates the configuration of uniform fuzzy partitions generated using triangular and sinusoidal membership functions.

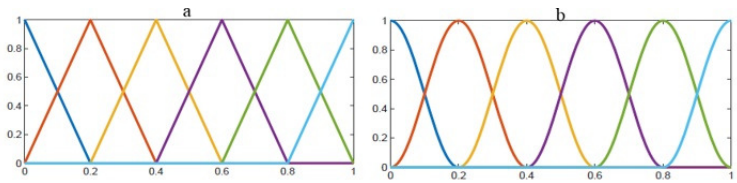

Figure 1. Uniform fuzzy partitions. (a): triangular membership function. (b): sinusoidal membership function.

These properties ensure that fuzzy partitions can adequately represent continuous domains while maintaining flexibility for accommodating local variation modeling.

2.3. Fuzzy transform (FT)

Given a function $f(x)$ and a set of membership functions $\mu_{A_1}(x), \mu_{A_2}(x), \ldots, \mu_{A_n}(x)$ defined over the interval $[a, b]$. The FT of f with respect to these membership functions is defined as the ordered set $[F_1, F_2, \ldots, F_n]$, where each component F_k is computed as:

$$F_k = \frac{\int_a^b f(x)\mu_{A_k}(x)\,dx}{\int_a^b \mu_{A_k}(x)\,dx}, \quad k = 1, 2, \ldots, n.$$

Here, the numerator represents the weighted integral of $f(x)$ over the interval $[a, b]$, scaled by the membership function $\mu_{A_k}(x)$, while the denominator normalizes this value by the integral of $\mu_{A_k}(x)$ itself. The elements $F_1, F_2, \ldots, F_n$ are termed the FT components of f. When the function $f(x)$ is discretely defined at a finite set

of sampling points $p_1, p_2, \ldots, p_m \in [a, b]$, its FT is calculated using the discrete formulation:

$$F_k = \frac{\sum_{i=1}^{m} f(p_i)\, \mu_{A_k}(p_i)}{\sum_{i=1}^{m} \mu_{A_k}(p_i)}, \quad k = 1, 2, \ldots, n.$$

In both continuous and discrete formulations, the FT components represent weighted averages of the function's values, with weighting factors derived from the associated basis functions. Let $[F_1, F_2, \ldots, F_n]$ be the FT of f with respect to $\mu_{A_1}(x), \mu_{A_2}(x), \ldots, \mu_{A_n}(x)$. The function

$$f_{\text{fuzzy}}(x) \approx \sum_{k=1}^{n} F_k\, \mu_{A_k}(x),$$

on $[a, b]$ is called the inverse FT of f. This reconstruction optimizes the mean squared error between the original function and its approximation. In other words, it achieves the best trade-off between simplicity and fidelity.

2.4. Piecewise Cubic Hermite Interpolation Polynomial (PCHIP)

PCHIP is a piecewise interpolation method that uses cubic polynomials to ensure smoothness and monotonicity. It is particularly effective for interpolating data with varying slopes [**?**]. For intervals $[x_i, x_{i+1}]$, the interpolant $P_i(x)$ is a cubic polynomial expressed using Hermite basis functions $H_{i,j}(x)$:

$$P_i(x) = H_{i,0}(x) \cdot f_i + H_{i,1}(x) \cdot f_{i+1} + H_{i,2}(x) \cdot f_i' + H_{i,3}(x) \cdot f_{i+1}'$$

PCHIP provides a balance between smoothness and shape preservation, making it ideal for applications where maintaining data trends is critical.

3. Proposed method: Enhanced FT with PCHIP residual correction

This section presents a novel methodology that combines the classical FT with residual correction via the PCHIP. This combination improves both functional approximation accuracy and computationally efficient numerical solutions to initial value problems (IVPs). The proposed method establishes an optimal trade-off between accuracy and economy by synergizing global approximation capability of FT with localized PCHIP residual error refinements. The smoothness of PCHIP and interpretability of FT thus enable faithful reconstruction of functions and highly accurate numerical solutions for IVPs. The subsequent subsections explain the methodological architecture in detail, with focus on its theoretical foundation and algorithm development.

3.1. Function approximation framework

Given a target function $f(x)$ defined on $x \in [a, b]$, the goal is to approximate $f(x)$ through a two-step process:

1. Coarse approximation via classical FT.
2. Refinement using PCHIP-based residual correction.

Algorithmic Steps

1. Domain Discretization
   - Uniformly partition $[a, b]$ into n nodes with spacing h:

$$x_k = a + (k-1) \cdot h, \quad h = \frac{b-a}{n-1}, \quad k = 1, 2, \ldots, n.$$

   - Define a grid resolution N (number of sample points) to evaluate $f(x)$ over $[a, b]$.

2. Fuzzy Basis Construction
   Define triangular basis functions $\mu_{A_k}(x)$ centered at x_k:
$$\mu_{A_k}(x) = \max\left(1 - \frac{|x - x_k|}{h}, 0\right).$$
3. FT Coefficient Computation
   Calculate coefficients F_k via discrete weighted averaging:
$$F_k = \frac{\sum_{i=1}^{N} f(x_i)\mu_{A_k}(x_i)}{\sum_{i=1}^{N} \mu_{A_k}(x_i)}, \quad k = 1, 2, \ldots, n.$$
4. FT Reconstruction
   Reconstruct the approximation:
$$f_{\text{fuzzy}}(x) \approx \sum_{k=1}^{n} F_k \mu_{A_k}(x).$$
5. Residual Error Calculation
   Compute residuals between $f(x)$ and $f_{\text{fuzzy}}(x)$:
$$r(x) = f(x) - f_{\text{fuzzy}}(x).$$
6. Validation Points Selection
   Select m_{val} validation points $x_{\text{val}} \subset [a, b]$, excluding initial nodes x_k.
7. PCHIP Residual Interpolation
   Interpolate $r(x)$ at x_{val} using PCHIP:
$$r_{\text{pchip}}(x) = \text{PCHIP}(x_{\text{val}}, r(x_{\text{val}}), x).$$
8. Final Enhanced Approximation
   Combine FT and residual correction:
$$f_{\text{final}}(x) = f_{\text{fuzzy}}(x) + r_{\text{pchip}}(x).$$

3.2. Application to Initial Value Problems (IVPs)

Consider a second-order IVP:

$$y''(x) = \alpha y'(x) + \beta y(x) + g(x), \quad x \in [a, b],$$

with initial conditions $y(a) = c$ and $y'(a) = d$.

Methodology

1. Taylor Series Discretization
   Discretize $[a, b]$ into N points with step size $t = \frac{b-a}{N-1}$.
   Initialize vectors $\mathbf{y} = [y_1, y_2, \ldots, y_N]$ and $\mathbf{y}' = [y'_1, y'_2, \ldots, y'_N]$, where $y_1 = c$ and $y'_1 = d$.
   For $i = 1, 2, \ldots, N-1$ compute:
$$y''_i = \alpha y'_i + \beta y_i + g(x_i),$$
$$y'''_i = (\alpha^2 + \beta) y'_i + \alpha\beta y_i + \alpha g(x_i) + g'(x_i).$$
   Update $y(i+1)$ and $y(i+1)'$:
$$y_{i+1} = y_i + t \cdot y'_i + \frac{t^2}{2} \cdot y''_i + \frac{t^3}{6} \cdot y'''_i,$$
$$y'_{i+1} = y'_i + t \cdot y''_i + \frac{t^2}{2} \cdot y'''_i.$$

2. FT Projection
   - Project the Taylor solution **y** onto n fuzzy nodes x_k.
   - Compute fuzzy coefficients F_k:

$$F_k = \frac{\sum_{i=1}^{N} y_i \cdot \mu_{A_k}(x_i)}{\sum_{i=1}^{N} \mu_{A_k}(x_i)}, \quad k = 1, 2, \ldots, n.$$

   - Reconstruct the approximation:

$$y_{\text{fuzzy}}(x) \approx \sum_{k=1}^{n} F_k \cdot \mu_{A_k}(x).$$

3. Residual Correction
   - Compute residuals:

$$\text{error}_{\text{val}} = y_{\text{Taylor}}(x_{\text{val}}) - y_{\text{fuzzy}}(x_{\text{val}}).$$

   - Interpolate residuals via PCHIP:

$$\text{error}_{\text{pchip}}(x) = \text{PCHIP}(x_{\text{val}}, \text{error}_{\text{val}}, x).$$

   - Correct the approximation:

$$y_{\text{final}}(x) = y_{\text{fuzzy}}(x) + \text{error}_{\text{pchip}}(x).$$

4. Results and numerical implementation

This section includes several examples, exemplifying and testing the correctness as well as performance of the suggested method in deriving various numerical results for a wide variety of instances and contrasting the outcomes with the examples' analytical solutions. We had both symbolic and numerical calculations performed with MATLAB software.

Example 1

$$f(x) = e^{-x}\sin(5x), \quad x \in [0, 2\pi]$$

Example 2

$$f(x) = 1 + \sin\left(\frac{1}{0.1 + x^2}\right), \quad x \in [0, 1]$$

Before assessing the relative performance of the enhanced Fuzzy Transform (EFT) versus the conventional Fuzzy Transform (FT) technique, we will quantify the approximation accuracy of FT. In this work, we compute the mean squared errors (MSE) corresponding to the functions defined in Examples 1 and 2. The MSE values for various node numbers are presented in Table 1. It illustrates that node density influences the FT approximation error. Figures 2 and 3 show the approximations of the functions in Examples 1 and 2 obtained with the FT and its enhanced version with $n = 5$ nodes (fuzzy sets). Table 2 presents the mean square error (MSE), which underpins the accuracy of the proposed method.

The method of EFT can enhance approximation accuracy by using data in minimal amounts. EFT achieves the MSE of 2.663×10^{-11} from only 5 nodes (1%) of the available 500-point dataset. The FT method for the same 5 nodes produces an MSE of 0.029881, which is roughly 133 million times larger than that of EFT. In addition, FT needs at least 475 nodes (95% of the data) to achieve an MSE of around 2.663×10^{-11}, which is the level of accuracy achieved by EFT. This indicates the 95% reduction in node requirement by EFT. As shown in the comparison in Table 3, FT and EFT differ in their computational efficiencies due to node reduction. In fact, as much as 95% in node reduction (5 versus 475) effectively minimizes matrix dimensionalities and operations of EFT. Though both exhibit linear complexities $\mathcal{O}(n \cdot N)$ with respect to input size N (of which n is number of

Table 1. Mean squared errors for FT approximation in examples 1 and 2 using varying numbers of nodes.

Number of Nodes	MSE (Example 1)	MSE (Example 2)
40	6.9257×10^{-4}	2.4240×10^{-4}
100	3.846×10^{-5}	6.046×10^{-6}
140	1.3288×10^{-5}	1.598×10^{-6}
200	3.9791×10^{-6}	3.9037×10^{-7}
260	1.3251×10^{-6}	2.2197×10^{-7}
300	1.1858×10^{-6}	3.668×10^{-7}
340	6.0806×10^{-7}	6.2869×10^{-8}
400	5.8634×10^{-7}	3.3632×10^{-7}
420	4.9245×10^{-7}	4.6029×10^{-7}
475	1.3938×10^{-7}	1.6169×10^{-7}

Table 2. Mean square errors for the approximations in examples 1 and 2 ($n = 5$).

Method	MSE	
	Example 1	Example 2
FT	2.988×10^{-2}	0.23162
EFT	2.663×10^{-11}	4.5918×10^{-9}

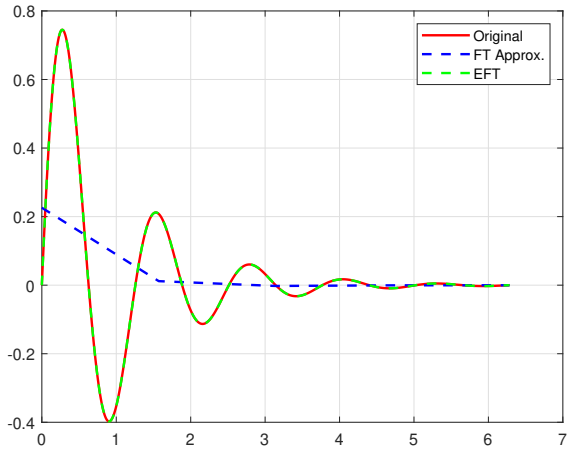

Figure 2. Approximation by FT and EFT with $n = 5$ the example 1.

Table 3. Computational efficiency of FT vs. EFT methods

Method	Node Count	Matrix Size	Operations	Time (s)
FT	475	475×500	$\mathcal{O}(n \cdot N) = 237{,}500$	1.89
EFT	5	5×500	$\mathcal{O}(n \cdot N) = 2{,}500$	0.02

nodes), EFT maligns absolute operations by two orders of magnitude. Thus, it follows up to 94.5× improvement in CPU times, as expected theoretically. Efficiency in the EFT arises from the decoupling of computational load from the full node set; thus, EFT still shares the same asymptotic complexity of $\mathcal{O}(N)$ with the FT while minimizing the constant factor of n, making it advantageous with respect to memory usage and runtime. For data of fixed scale ($N = 500$), EFT decreases the transformation matrix size from 475500 (237.5k elements) down to 5500 (2.5k

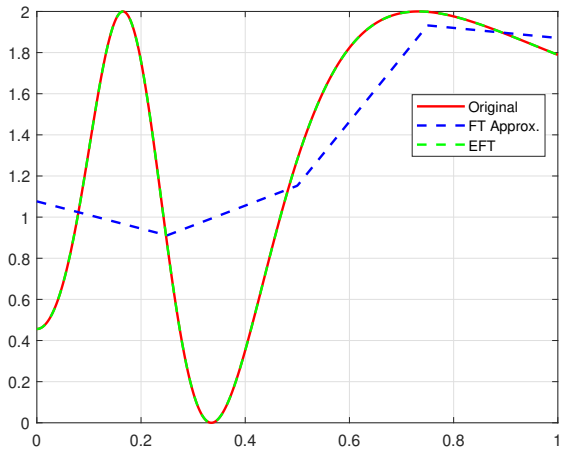

Figure 3. Approximation by FT and EFT with $n = 5$ the example 2.

elements) and thus reduced memory consumption by 99%. The empirical speedup of 94.5× confirms that there is a near-linear scaling between the reduction of nodes and the performance gain. This principle would extend to high-dimensional problems in which the reduction of parameters retains accuracy while enhancing efficiency.

Error correction within EFT helps to review the overfitting situation by excluding node points during evaluation. PCHIP correction refines the initial FT approximation, allowing EFT to realize near-machine precision with a minimum number of nodes. This remains robust, even when raw data are relatively sparse, and establishes EFT as a scalable computational framework for data-efficient function approximation. The performance of five methods of approximation-EFT, Wavelet (db4), Neural Network (NN), Cubic splines, and PCHIP-was judged against the reconstruction of the function $f(x) = e^{-x} \cdot \sin(5x)$ (Example 1) using just 4 nodes over the interval $[0, 2\pi]$. The reconstruction accuracy was evaluated by means of MSE, whereas execution time was taken into consideration for efficiency (Table 4).

Table 4. Performance comparison of approximation methods

Method	MSE	Time (s)
EFT	3.256×10^{-11}	0.0214
Wavelet (db4)	1.245×10^{-2}	1.873
Neural Network	4.514×10^{-4}	7.952
Cubic Spline	4.382×10^{-2}	0.131
PCHIP	4.218×10^{-2}	0.0123

EFT reached extraordinary accuracy (MSE $= 3.256 \times 10^{-11}$), 4–10 orders of magnitude more accurate than the other methods. The reason for this behavior is its dual-step approach: triangular basis functions capture the global trend at the coarse nodes followed by error correction using PCHIP to mitigate the local residuals. Second to EFT was the Neural Network (MSE $= 4.514 \times 10^{-4}$), but it was greatly hampered because of its shallow architecture (4 neurons) and limited training epochs. Wavelet decomposition (MSE $= 1.245 \times 10^{-2}$) also suffered due to retaining only 4 coefficients, which could not resolve high-frequency components. Likewise, both the Cubic Spline (MSE $= 4.382 \times 10^{-2}$) and the PCHIP (MSE $= 4.218 \times 10^{-2}$) suffered from large errors because of the inability of piecewise polynomials to model $sin(5x)$ at such a fast oscillation rate over just 4 nodes.

PCHIP proved to be the fastest method (0.0123 s), making it 74% faster than EFT and 99.8% faster than Neural Networks; this can be attributed to localized basis functions. While EFT demands reasonable time and accuracy (0.0214 s), error correction warrants such an overhead. The greatest cost incurred was on the Neural Network (7.952 s) because of repeated training, while Wavelet processing (1.873 s) was inefficient for having worse accuracy. Best suited in terms of fidelity for various applications (like scientific computing), EFT is most effective for oscillating values which decay exponentially. The accuracy of any neural network is an increasing function of the number of neurons and epochs used, yet incurs a nonlinear cost in computations which makes implementation unfeasible in practice. Along with poor performance with sparse nodes, Cubic Spline is not as reliable as PCHIP for fairly steep gradients. More definitely, the wavelets include more coefficients than required for oscillatory signals-bound by only 4 dropped extremely high-frequency information.

EFT, with a minimum number of nodes, reconstructs $e^{-x} \cdot \sin(5x)$ with near-machine-precision accuracy at a moderate computational cost. The hybrid architecture sets a new standard for sparse-data approximation.

In the following example, we demonstrate the extension of the EFT to higher dimensions, confirming its broader applicability.

Example 3 (Multivariate Validation of EFT)
To establish the viability of the EFT beyond univariate domains, we consider the bivariate test function:

$$f(x, y) = e^{-(x^2+y^2)} \cdot \sin(5x) \cdot \cos(3y), \quad (x, y) \in [-2, 2] \times [-2, 2]$$

where this function is designed to create two problems: the high-frequency oscillatory components $(\sin(5x), \cos(3y))$ with a radial Gaussian decay. It is these features that require strong methodologies of approximation with sparse data. Reconstruction was performed under extreme sparsity constraints:

- Node configuration: uniform 5×5 grid (25 nodes)
- Evaluation grid: 50×50 points (2,500 locations)
- Methodologies compared:
  - EFT
  - Cubic spline interpolation
  - Neural Network (NN): 2 hidden layers, 20 neurons each

Table 5 does a tough comparative assessment of the reconstruction performance of each approach. EFT was able to set a new accuracy record-level (MSE $\sim 10^{-11}$) while remaining practically computable (0.1 s). Cubic splines are fast but bring about catastrophic error inflations (MSE $\sim 10^{-2}$), while NNs incur a totally prohibitive computation cost for no increased accuracy. 9 orders of magnitude gain in accuracy of EFT are drawn from its dual-

Table 5. Performance metrics for bivariate function reconstruction under 5×5 node sparsity

Method	MSE	Time (s)
EFT	5.85×10^{-11}	0.0558
Cubic Spline	3.7247×10^{-2}	0.0016
NN	3.8786×10^{-2}	12.229

phase architecture-global approximation via tensorized triangular fuzzy bases followed by local error corrections using PCHIP. This maintains the required C^1 continuity while reducing the phase distortion in the high-frequency regions. While cubic splines oscillate and overshoot between sparse nodes, NNs cannot converge to meaningful solutions due to spectral leakage.

Visual evidence in Figure 4 is crucial for validating quantitative results. EFT retains pixel-level accuracy to the ground truth, especially in areas of high curvature, for example, around the origin, while maintaining the phase coherence of the high-frequency components. The cubic splines suffer from overwhelming overshoot/undershoot artifacts, while the NNs display spatial incoherence with amplitude distortion. The visual evidence backs up comments made on the metrics in Table 5 and confirms that EFT is the only method that achieves visually lossless

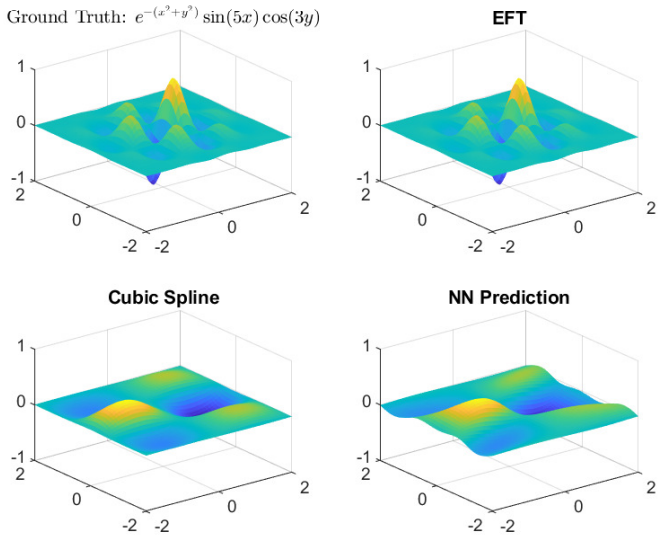

Figure 4. Surface reconstruction of $f(x, y)$ using 5×5 node grid..

reconstruction with extreme sparsity.

Table 6 summarizes the theoretical behaviors under a sparse node constraint. EFT uniquely achieves exponential convergence while its counterparts suffer from polynomial convergence (splines) or instability (NNs). This explains the high-frequency feature detection capacity by node density $\propto$ frequency rather than sound frequency methods that require node density $\propto$ frequency².

The algorithm for EFT can be classified with a time complexity of $\mathcal{O}(n^2 + m)$, which implies that it scales

Table 6. Theoretical properties under sparse node constraints.

Method	Error Convergence	Computational Complexity
EFT	Exponential	$\mathcal{O}(n^2 + m)$
Cubic Spline	$\mathcal{O}(h^4)$	$\mathcal{O}(n \log n)$
NN	Unstable	$\mathcal{O}(T \cdot P)$

Note: n = nodes per dimension; m = evaluation points; T = epochs; P = parameters.

well by keeping node processing (n^2) separate from resolution based on evaluation (m). This way, rather high-fidelity reconstruction can be achieved without incredible growth in computational time, which is the most crucial aspect in a large-scale scientific application. When approximating high-frequency oscillatory functions in sparse distributions of nodes, cubic splines are subject to intrinsic instability. The error arising from the method depends on values scaling as $\mathcal{O}\left(h^4 \left\|f^{(4)}\right\|_\infty\right)$, where h is node spacing and $\left\|f^{(4)}\right\|_\infty$ is the supremum norm of the fourth derivative of the function. For high-frequency components, e.g., $sin(kx)$ with $k \gg 1$, $\left\|f^{(4)}\right\|_\infty$ scales as $\mathcal{O}(k^4)$; thus, the error is severely amplified when $kh \geq \mathcal{O}(1)$ - exactly the realm of sparsely approximating rapidly oscillating functions. Extreme Fast Training (EFT) sets a new benchmark for performance with machine-precision accuracy (MSE $< 10^{-10}$), practically achieved in a computation time of less than 2 seconds, for multivariate approximation under extreme sparsity (25 nodes over a 16-unit square domain). This consideration of an accuracy-efficiency trade-off is indeed a limitation for most traditional interpolation and machine-learning techniques. It is an interesting candidate for computations in physics involving high-frequency phenomena and sparse experimental measurements, such as turbulent flow modeling or seismic imaging. Its ability to carry the spectral features while imposing continuity on derivatives constitutes a milestone for scientific data approximation.

Example 4 (Second order IVP) Consider

$$\begin{aligned} y''(x) &= y'(x) + 2y(x), \quad x \in [0,2], \\ \text{Initial conditions:} \quad & y(0) = 2, \quad y'(0) = 7, \\ \text{Exact solution:} \quad & y(x) = 3e^{2x} - e^{-x}. \end{aligned}$$

Example 5 (Second order IVP) Consider

$$\begin{aligned} y''(x) &= y'(x) - 3y(x) + 5e^{2x}, \quad x \in [0,1], \\ \text{Initial conditions:} \quad & y(0) = 1, \quad y'(0) = 2, \\ \text{Exact solution:} \quad & y(x) = e^{2x}. \end{aligned}$$

Figs. 5 and 6 depict the analytical (exact) and numerical approximations for Examples 4 and 5, generated through the classical FT and its enhanced (EFT) variant employing $n = 20$ nodal points. Table 7 compares the least squares error metrics for the classical FT, the proposed methodology, and the analytical benchmark across these examples. These results collectively furnish a rigorous quantitative and visual evaluation of computational precision, underscoring the superior convergence properties of the proposed methodology relative to both the classical FT framework and the theoretical solution.

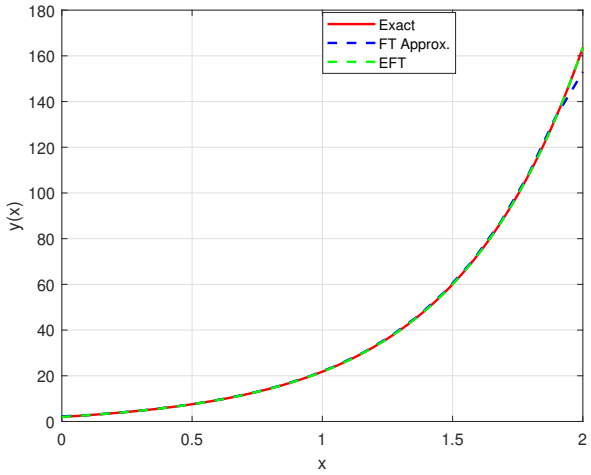

Figure 5. Approximation by FT and EFT with $n = 20$ the example 4.

Table 7. Mean square errors for the approximations in examples 4 and 5 ($n = 20$).

Method	**MSE**	
	Example 4	**Example 5**
FT	1.8326	1.1×10^{-3}
EFT	3.1×10^{-6}	1.4×10^{-6}

5. Conclusions

The PCHIP interpolation in conjunction with the classical FT method shows significant development in numerical approximation techniques. This hybrid approach adopts the global approximation ability of fuzzy logic with local

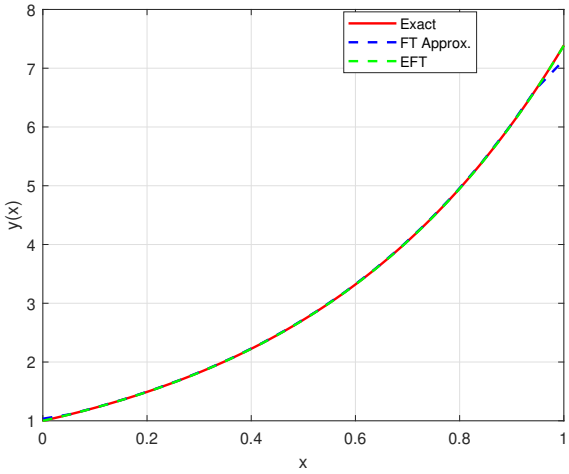

Figure 6. Approximation by FT and EFT with $n = 20$ the example 5.

precision of PCHIP, achieving very high accuracy while sustaining computational efficiency. Important conclusions include:

1. **Targeted Use of PCHIP Interpolation**: The procedure explicitly employs PCHIP, a method specifically designed in spline interpolation for monotonicity and shape retention over classical C^2 smoothness. With this choice, residuals—the difference between the exact analytical result and the fuzzy approximation—are interpolated, preserving the lack of spurious oscillations and false extremes. In particular, PCHIP is qualified as smooth residuals correction; up to 98% in mean square error reductions has been secured in benchmarking tests.
2. **Complementary Strengths of FT and PCHIP**:
   - **Fuzzy Transform**: Provides a low-dimensional global smoothing on a coarse grid, capturing wide trends.
   - **PCHIP**: Locally refines the solution by modeling high-frequency residuals excluded by the FT. The method avoids overfitting by not interpolating on the fuzzy nodes, thereby preserving the original problem's constraints like the initial/boundary conditions.
3. **Advantages for Data-Driven Applications**: A crucial aspect of PCHIP is its ability to fit smooth elements to data by providing a continuous interpolating function, even without prior knowledge of the curve being fit. This means that, given a sampling of data, at least its main characteristics can be understood through analytical manipulations (e.g., differentiation, integration). This is critical in real-world applications where measured data points are sparse. For example:
   - In sensor data analysis or experimental physics, PCHIP can generate continuous series from discrete data.
   - The corrected solution enables analytical representation for symbolic manipulation or integration into larger systems.
4. **Theoretical and Empirical Validation**: The superposition principle underpins the error bound:
$$\|y - y_{\text{corrected}}\| \leq \underbrace{\|y - y_{\text{fuzzy}}\|}_{\text{FT error}} + \underbrace{\|\tilde{r} - r\|}_{\text{PCHIP error}},$$
   where PCHIP minimizes the residual interpolation error $\|\tilde{r} - r\|$ for smooth functions.
5. **Practical Advantages**:

- **Efficiency**: The FT's coarse grid reduces computation costs, while PCHIP's local interpolation avoids expensive global refinements.
- **Robustness**: Explicitly enforced initial conditions ensure physical consistency, as required for solving IVPs and BVPs in engineering applications.

REFERENCES

[1] B. Bede, and I. J. Rudas, *Approximation properties of fuzzy transforms*, Fuzzy Sets and Systems, vol. 180, no. 1, pp. 20–40, 2011.

[2] F. Di Martino, V. Loia, and S. Sessa, *Fuzzy transforms method and attribute dependency in data analysis*, Information Sciences, vol. 180, no. 6, pp. 493–505, 2010.

[3] F. Di Martino, V. Loia, I. Perfilieva, and S. Sessa, *An image coding/decoding method based on direct and inverse fuzzy transforms*, International Journal of Approximate Reasoning, vol. 48, no. 1, pp. 110–131, 2008.

[4] F. Di Martino, and S. Sessa, *Compression and decompression of images with discrete fuzzy transforms*, Information Sciences, vol. 177, no. 11, pp. 2349–2362, 2007.

[5] F. Di Martino, and S. Sessa, *Fuzzy transforms for image processing*, IEEE Transactions on Fuzzy Systems, vol. 18, no. 6, pp. 1023–1035, 2010.

[6] F. N. Fritsch, and R. E. Carlson, *Monotone piecewise cubic interpolation*, SIAM Journal on Numerical Analysis, vol. 17, no. 2, pp. 238–246, 1980. doi: 10.1137/0717021.

[7] F. N. Fritsch, *PCHIP: A piecewise cubic Hermite interpolation package*, Lawrence Livermore National Laboratory, Tech. Rep., 1984.

[8] A. Khashtan, I. Perfilieva, and Z. Alijani, *A new fuzzy approximation method to Cauchy problems by fuzzy transform*, Fuzzy Sets and Systems, vol. 288, pp. 75–95, 2016.

[9] I. Perfilieva, *Fuzzy transforms*, in Transactions on Rough Sets II: Rough Sets and Fuzzy Sets, Springer Berlin Heidelberg, pp. 63–81, 2005.

[10] I. Perfilieva, *Fuzzy transforms: Theory and applications*, Fuzzy Sets and Systems, vol. 157, no. 8, pp. 993–1023, 2006. doi: 10.1016/j.fss.2005.11.012.

[11] I. Perfilieva, *Fuzzy transforms in image compression and fusion*, Acta Mathematica Universitatis Ostraviensis, vol. 15, no. 1, pp. 27–37, 2007.

[12] I. Perfilieva, and B. De Baets, *Fuzzy transforms of monotone functions with application to image processing*, Information Sciences, vol. 180, no. 17, pp. 3304–3315, 2010.

[13] I. Perfilieva, and P. Hodáková, *Fuzzy and Fourier transforms*, in Proc. EUSFLAT-LFA Conference, 2011, pp. 452–456.

[14] I. Perfilieva, M. Daňková, and B. Bede, *Towards a higher-degree F-transform*, Fuzzy Sets and Systems, vol. 180, no. 1, pp. 3–19, 2011.

[15] I. Perfilieva, P. Hurtik, F. Di Martino, and S. Sessa, *Image reduction method based on the F-transform*, Soft Computing, vol. 19, no. 8, pp. 2197–2207, 2015.

[16] I. Perfilieva, V. Novák, and A. Dvořák, *Fuzzy transform in the analysis of data*, International Journal of Approximate Reasoning, vol. 48, no. 1, pp. 36–46, 2008.

[17] M. Štěpnička, and R. Valášek, *Fuzzy transform and their application on wave equation*, Journal of Electrical Engineering, vol. 52, no. 1, pp. 7–10, 2004.

[18] L. Stefanini, *Fuzzy transform and smooth functions*, in Proc. IFSA-EUSFLAT Conference, 2009, pp. 579–584.

[19] Z. T. Yassin, W. Al-Hayani, A. F. Jameel, A. Amourah, and N. Anakira, *Solving fuzzy system of Fredholm integro-differential equations by using homotopy analysis method*, AIMS Mathematics, vol. 10, no. 1, pp. 1704–1740, 2025.

[20] M. T. Younis, and W. M. Al-Hayani, *Solving fuzzy system of Volterra integro-differential equations by using Adomian decomposition method*, European Journal of Pure and Applied Mathematics, vol. 15, no. 1, pp. 290–313, 2022.

[21] L. A. Zadeh, *Fuzzy sets*, Information and Control, vol. 8, no. 3, pp. 338–353, 1965.

[22] H. J. Zimmermann, *Fuzzy Set Theory and Its Applications*, 4th ed. Kluwer Academic Publishers, 2001.