Gulf Journal of Mathematics
Vol 22, Issue 2 (2026) 1-16

ISSN: 2309-4966
https://doi.org/10.56947/gjom.v22i2.4041

FRACNN: FRACTIONAL SPLINE NEURAL NETWORKS

AMAL BOUAICHA[1], MOHAMED LAMNII[1] and SOUMIA NGADI[1*]

Abstract. We propose a new neural architecture inspired by the structural principles of Kolmogorov-Arnold theorems, in which fixed activation functions are replaced by learnable fractional splines. Unlike classical approaches, our model deals not only with weights and biases as trainable parameters, but also with fractional orders and shape parameters governing the activation functions themselves. The network's local regularity can be dynamically adapted during learning, which favors sharper responses in areas that need detailed sharpness, and smoother transitions where stability or generalization is of importance.We derive exact analytical gradients for all parameters including fractional orders, which makes possible end-to-end optimization via gradient-based methods. The theoretical foundation is based on the constructive properties of fractional spline bases and their ability to provide an adaptive, multi-scale functional approximation. Our approach thus establishes a bridge between geometric flexibility and differentiable learning, providing a theory-based framework for designing data-driven activation functions.

Keywords. Fractional splines, Kolmogorov-Arnold networks, adaptive activation functions, neural networks , deep learning
2020 Mathematics Subject Classification. Primary 46L55; Secondary 44B20

1. Introduction and preliminaries

1.1. **Introduction.** In the field of machine learning, artificial neural networks are widely used to model complex functions from data. However, their structure often relies on fixed activation functions like ReLU or tanh that never change during learning, even though the data varies considerably from one problem to another.

Our work is inspired by the approach proposed by [11] in their article titled *KAN: Kolmogorov-Arnold Networks*, where they introduce a neural architecture based on the Kolmogorov-Arnold, a fundamental mathematical result stating that a continuous multivariate function can be represented as a finite sum of functions composed of a single variable [10, 2].

Unlike classical networks, where activation functions are fixed a priori and never change during learning, our model replaces these functions with fractional causal

Date: Received: Dec 31, 2026; Revised: Jan 28, 2026; Accepted: Feb 9, 2026.
* Corresponding author

splines, the shape of which is adjusted automatically during training. While traditional approaches are limited to optimizing weights and biases, our architecture also allows adjusting the shape parameters of splines, thus offering direct control over the local model list.

This approach extends that of the KAN by replacing learnable univariate functions with splines whose shape parameters are optimized, which allows both to adjust the local complexity of the model and to guarantee a certain mathematical regularity. Each network connection thus corresponds to a well-defined spline, whose shape is controlled by an interpretable parameter. The network then becomes a transparent tool, capable not only of predicting, but also of explaining how it has learned to represent data, a crucial requirement in scientific, medical or engineering fields.

Several recent works show that splines, in particular fractional splines, can improve the accuracy and numerical stability in solving differential equations. For example, [5] showed that quintic splines allow for lower numerical errors and higher convergence rates than classical polynomial methods. Similarly, [14] demonstrated that finite elements based on quadratic splines offer better stability and faster convergence than Lagrangian elements for cited convection-diffusion.

Another advantage of our approach is that it allows for a better understanding of the model's learning mechanism. Each neuron corresponds to a well-defined spline, whose shape is controlled by an interpretable parameter. This reinforces the explicability of the network, which is no longer limited to predictive performance, but also provides an intelligible representation of the learning process.

1.2. **Mathematical Foundations: Fractional Splines.** Fractional splines are a natural generalization of classical polynomial splines [4]. They extend spline theory to non integer orders $\alpha > -1$. This extension is based on fractional calculus and gives continuous control over smoothness, approximation order, and frequency behavior [17]. In this work, we establish the mathematical foundations of these functions. We focus on their construction using fractional finite differences, their optimal approximation properties, and their link to wavelets [17]. We compare their performance with classical B-splines, highlighting their advantages for interpolating non smooth or fractal data. We also discuss challenges in their numerical implementation [1]. Our analysis shows that fractional splines form a rich and flexible family, with strong potential for applications in numerical analysis, signal processing, and solving fractional differential equations [13].

1.2.1. *Construction of Fractional Splines.*

Definition 1.1. Let $\alpha > -1$ be a real number. The one-sided power function of order α is defined as:

$$x_+^\alpha = \begin{cases} x^\alpha & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

This function is used to build fractional splines. It is important in numerical approximation and fractional calculus [17].

Definition 1.2. For a function f, the fractional finite difference operator of order α is defined by:

$$\Delta_+^\alpha f(x) = \sum_{k=0}^{\infty} (-1)^k \binom{\alpha}{k} f(x-k),$$

where $\binom{\alpha}{k}$ is the generalized binomial coefficient:

$$\binom{\alpha}{k} = \frac{\Gamma(\alpha+1)}{\Gamma(k+1)\Gamma(\alpha-k+1)}.$$

This operator extends the idea of finite differences to non-integer orders. It is key to building fractional splines [17].

Definition 1.3. By analogy with classical B-splines, the causal fractional B-spline of order $\alpha > -1$ is defined as:

$$\beta_+^\alpha(x) := \frac{1}{\Gamma(\alpha+1)} \Delta_+^{\alpha+1} x_+^\alpha = \frac{1}{\Gamma(\alpha+1)} \sum_{k=0}^{\infty} (-1)^k \binom{\alpha+1}{k} (x-k)_+^\alpha.$$

This is the $(\alpha+1)$-th fractional difference of the function x_+^α. This formula is the basic definition of fractional splines and works for all $\alpha > -1$ [17].

Figure 1 illustrates how the causal fractional B-spline $\beta_+^\alpha(x)$ changes when the parameter α increases from 0 to 5. When $\alpha = 0$, the spline behaves like a step function. For $\alpha > 0$, it becomes continuous and progressively smoother. Its shape is compact and rounded, and it becomes wider and flatter as α increases. This behavior shows the smoothing and interpolation properties of fractional splines, where the level of smoothness is directly controlled by the parameter α.

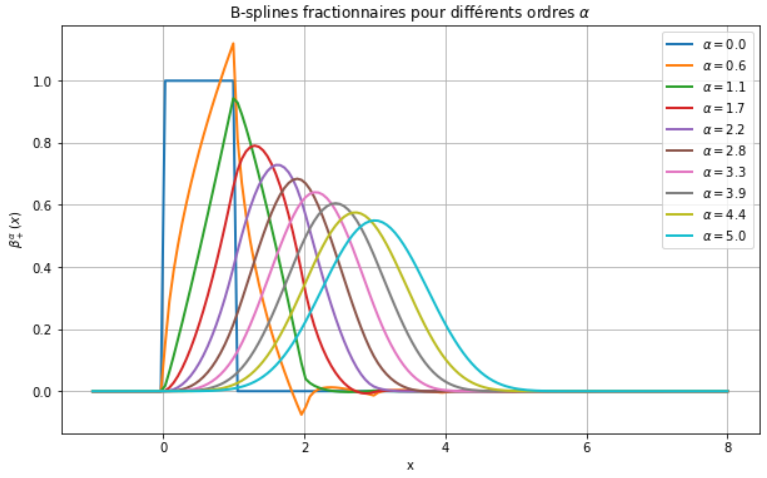

FIGURE 1. Causal fractional B-splines $\beta_+^\alpha(x)$ for different orders α.

Partial Derivative of a Causal Fractional B-Spline with Respect to its Order α: The partial derivative of $\beta_+^\alpha(x)$ with respect to α is obtained by applying the product rule to its infinite series representation. Each term in the sum depends on α, so we write:

$$\frac{\partial \beta_+^\alpha}{\partial \alpha}(x) = \left[\frac{\partial}{\partial \alpha}\left(\frac{1}{\Gamma(\alpha+1)}\right)\right]\cdot\sum_{k=0}^{\infty}(-1)^k\binom{\alpha+1}{k}(x-k)_+^\alpha + \frac{1}{\Gamma(\alpha+1)}\cdot\frac{\partial}{\partial \alpha}\left[\sum_{k=0}^{\infty}(-1)^k\binom{\alpha+1}{k}(x-k)_+^\alpha\right].$$

We compute the two terms separately below.

Step 1: Derivative of the Factor $\frac{1}{\Gamma(\alpha+1)}$.

$$\frac{\partial}{\partial \alpha}\left(\frac{1}{\Gamma(\alpha+1)}\right) = -\frac{\Gamma'(\alpha+1)}{[\Gamma(\alpha+1)]^2}.$$

So the first term becomes:

$$T_1 = -\frac{\Gamma'(\alpha+1)}{[\Gamma(\alpha+1)]^2}\cdot\sum_{k=0}^{\infty}(-1)^k\binom{\alpha+1}{k}(x-k)_+^\alpha = -\frac{\Gamma'(\alpha+1)}{\Gamma(\alpha+1)}\beta_+^\alpha(x).$$

Step 2: Derivative of the Sum $\sum_{k=0}^{\infty}(-1)^k\binom{\alpha+1}{k}(x-k)_+^\alpha$.
Under the assumption of uniform convergence, we can interchange differentiation and summation:

$$\frac{\partial}{\partial \alpha}\left[\sum_{k=0}^{\infty}(-1)^k\binom{\alpha+1}{k}(x-k)_+^\alpha\right] = \sum_{k=0}^{\infty}(-1)^k\frac{\partial}{\partial \alpha}\left[\binom{\alpha+1}{k}(x-k)_+^\alpha\right].$$

Each term in the sum is a product of two functions of α. Using the product rule:

$$\frac{\partial}{\partial \alpha}\left[\binom{\alpha+1}{k}(x-k)_+^\alpha\right] = \left[\frac{\partial}{\partial \alpha}\binom{\alpha+1}{k}\right]\cdot(x-k)_+^\alpha + \binom{\alpha+1}{k}\cdot\frac{\partial}{\partial \alpha}\left[(x-k)_+^\alpha\right].$$

We compute these two partial derivatives below.

Derivative of $\binom{\alpha+1}{k}$:
Using the Gamma function extension, we have:

$$\binom{\alpha+1}{k} = \frac{\Gamma(\alpha+2)}{\Gamma(k+1)\,\Gamma(\alpha+2-k)}.$$

Since k is fixed, the term $\Gamma(k+1)$ does not depend on α. Let us define the constant

$$C_k := \frac{1}{\Gamma(k+1)},$$

so we can write

$$\binom{\alpha+1}{k} = C_k\,\frac{\Gamma(\alpha+2)}{\Gamma(\alpha+2-k)}.$$

Let

$$f(\alpha) := \binom{\alpha+1}{k}.$$

For any strictly positive function f, the logarithmic derivative gives:

$$\frac{d}{d\alpha}\ln f(\alpha) = \frac{f'(\alpha)}{f(\alpha)}, \quad f'(\alpha) = f(\alpha)\,\frac{d}{d\alpha}\ln f(\alpha).$$

Taking the logarithm of $f(\alpha)$, we get:

$$\ln f(\alpha) = \ln C_k + \ln\Gamma(\alpha+2) - \ln\Gamma(\alpha+2-k).$$

The derivative of the constant $\ln C_k$ is zero, so:

$$\frac{d}{d\alpha}\ln f(\alpha) = \frac{d}{d\alpha}\ln\Gamma(\alpha+2) - \frac{d}{d\alpha}\ln\Gamma(\alpha+2-k).$$

The digamma function ψ is defined for $z>0$ by:

$$\psi(z) := \frac{d}{dz}\ln\Gamma(z) = \frac{\Gamma'(z)}{\Gamma(z)}.$$

By the chain rule:

$$\frac{d}{d\alpha}\ln\Gamma(\alpha+2) = \psi(\alpha+2), \quad \frac{d}{d\alpha}\ln\Gamma(\alpha+2-k) = \psi(\alpha+2-k).$$

Thus,

$$\frac{d}{d\alpha}\ln f(\alpha) = \psi(\alpha+2) - \psi(\alpha+2-k).$$

Finally,

$$\frac{\partial}{\partial\alpha}\binom{\alpha+1}{k} = \binom{\alpha+1}{k}\left[\psi(\alpha+2) - \psi(\alpha+2-k)\right].$$

Derivative of $(x-k)_+^\alpha$:
For $x>k$, we have $(x-k)_+^\alpha = (x-k)^\alpha$, so:

$$\frac{\partial}{\partial\alpha}\left[(x-k)_+^\alpha\right] = (x-k)^\alpha\ln(x-k) = (x-k)_+^\alpha\ln(x-k)_+.$$

For $x\le k$, the function $(x-k)_+^\alpha$ is identically zero, so its derivative is also zero.
Thus, for all x:

$$\frac{\partial}{\partial\alpha}\left[(x-k)_+^\alpha\right] = (x-k)_+^\alpha\ln(x-k)_+.$$

Combining the results above, we get:

$$\frac{\partial}{\partial\alpha}\left[\binom{\alpha+1}{k}(x-k)_+^\alpha\right] = \binom{\alpha+1}{k}(x-k)_+^\alpha\left[\psi(\alpha+2) - \psi(\alpha+2-k) + \ln(x-k)_+\right].$$

The sum over k becomes:

$$\sum_{k=0}^{\infty}(-1)^k\frac{\partial}{\partial\alpha}\left[\binom{\alpha+1}{k}(x-k)_+^\alpha\right] = \sum_{k=0}^{\infty}(-1)^k\binom{\alpha+1}{k}(x-k)_+^\alpha\left[\psi(\alpha+2) - \psi(\alpha+2-k) + \ln(x-k)_+\right]$$

The second term of the total derivative is therefore [15]:

$$T_2 = \frac{1}{\Gamma(\alpha+1)}\sum_{k=0}^{\infty}(-1)^k\binom{\alpha+1}{k}(x-k)_+^\alpha\left[\psi(\alpha+2) - \psi(\alpha+2-k) + \ln(x-k)_+\right].$$

Adding T_1 and T_2, we obtain the full expression for the partial derivative:

$$\begin{aligned}\frac{\partial \beta_+^\alpha}{\partial \alpha}(x) = &- \frac{\Gamma'(\alpha+1)}{\Gamma(\alpha+1)}\beta_+^\alpha(x) \\ &+ \frac{1}{\Gamma(\alpha+1)}\sum_{k=0}^{\infty}(-1)^k\binom{\alpha+1}{k}\left[\psi(\alpha+2)-\psi(\alpha+2-k)\right](x-k)_+^\alpha \\ &+ \frac{1}{\Gamma(\alpha+1)}\sum_{k=0}^{\infty}(-1)^k\binom{\alpha+1}{k}(x-k)_+^\alpha \ln(x-k)_+.\end{aligned}$$

This expression decomposes how sensitive the causal fractional B-spline $\beta_+^\alpha(x)$ is to changes in its order α, using only the Gamma function and its derivative. It is especially useful in numerical applications where α is optimized, such as in interpolation or solving fractional differential equations [19].

Figure 2 shows graphically $\beta_+^{2.5}(x)$ (blue curve) and its derivative with respect to α, $\partial_\alpha\beta_+^{2.5}(x)$ (orange dashed curve). We observe strong oscillations around the integer points $x = 1, 2, 3$, reflecting the local sensitivity of the spline to changes in α in these regions. This property is crucial for optimization: it tells us where a small change in α will have the biggest impact on the curve's shape. This is useful for fine tuning smoothness or support localization in interpolation or modeling applications [20].

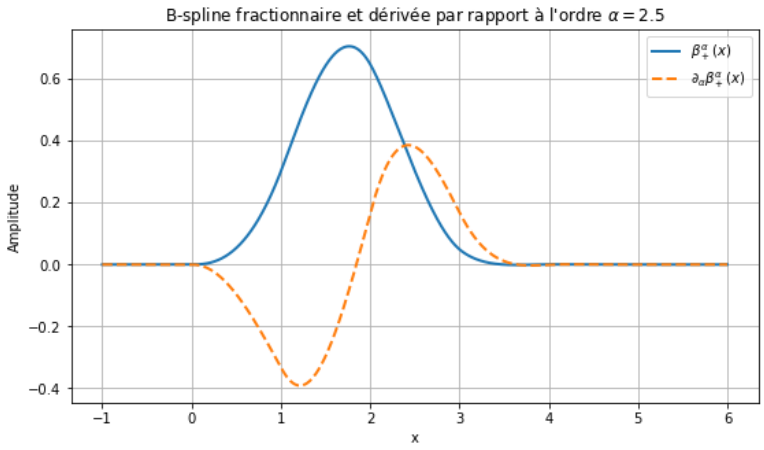

FIGURE 2. Causal fractional B-spline $\beta_+^{2.5}(x)$ and its partial derivative with respect to the order α

1.3. **Kolmogorov-Arnold Networks: Conceptual Framework.** This section explains why we use Kolmogorov-Arnold Networks (KAN) as a conceptual and mathematical alternative to classical Multilayer Perceptrons (MLP). The choice of KAN is based on strong theory, better functional interpretability, and a clear separation between linear and nonlinear parts. This is especially useful in contexts where smoothness and analytical control of models are important [11].

The Kolmogorov-Arnold representation theorem [11] says that any continuous function

$$f : [0,1]^n \longrightarrow \mathbb{R}$$

can be written as a finite sum of compositions of univariate continuous functions. More precisely, there exist continuous univariate functions $\phi_{q,p}$ and ψ_q such that

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \psi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right).$$

This fundamental result shows that high dimensional complexity can be broken down into a combination of one dimensional transformations. It gives a solid theoretical basis for approximating functions in many dimensions [10, 2].

Functional Interpretation and KAN Architecture.
Inspired by this theorem, the architecture of Kolmogorov-Arnold Networks uses a functional form like:

$$f(x) = \sum_j c_j \, \phi_j\big(w_j^\top x + b_j\big) \,,$$

where each term involves a linear transformation of the input x via weights $w_j \in \mathbb{R}^n$ and bias $b_j \in \mathbb{R}$, followed by a univariate basis function ϕ_j that may be nonlinear and parametrized, and finally scaled by a coefficient $c_j \in \mathbb{R}$. Unlike classical MLPs, the nonlinearity is not fixed or uniformly applied across neurons; instead, it resides in the functions ϕ_j, which are often selected for their favorable mathematical properties such as smoothness, differentiability, and locality [11].

Key Differences with Multilayer Perceptrons.
The main difference between KAN and MLP is how they handle nonlinearity. In an MLP, every neuron uses the same fixed activation function, like ReLU or tanh. This makes it hard for the model to adapt to different parts of the data. In a KAN, each connection has its own function ϕ_j, which can be learned during training. This lets the model adjust its behavior locally making it more flexible and easier to understand.
Multilayer perceptrons (MLPs) are characterised by the implementation of numerous layers of non linear functions. This methodology is effective for complex tasks; however, it obscures the internal workings of the model. In contrast, KANs provide a distinct formula, comprising a sum of elementary functions. The process of transformation of the input at each stage is rendered visible. This facilitates the comprehension of researchers in relation to both the output and the methodology employed by the model to derive that result [11].

2. Proposed Architecture:

This model extends classical Kolmogorov-Arnold networks by replacing standard activation functions with *fractional causal B-splines*, whose smoothness is controlled by a learnable real-valued order parameter $\alpha > 0$. The goal is to combine the expressive power of neural networks with the mathematical transparency and regularity control of spline based approximation [11].

The architecture preserves the core principle of Kolmogorov-Arnold decomposition: separating linear transformations from nonlinear activations, while introducing adaptive, differentiable basis functions that can be optimized during training [17].

2.1. **Global Matrix Formulation.** Let $X \in \mathbb{R}^{N\times d}$ denote the input matrix, where each row $x_i^\top \in \mathbb{R}^d$ represents one sample, and N is the total number of samples. The first layer performs an affine transformation:

$$Z = XA + \mathbf{1}_N b^\top,$$

where $A \in \mathbb{R}^{d\times m}$ is the weight matrix mapping inputs to m latent features, $b \in \mathbb{R}^m$ is the bias vector, $\mathbf{1}_N \in \mathbb{R}^N$ is the column vector of ones (i.e., $\mathbf{1}_N = [1, 1, \ldots, 1]^\top$), and $Z \in \mathbb{R}^{N\times m}$ is the resulting matrix of pre-activations. This step is purely linear: it projects the input data into a higher dimensional space without introducing any nonlinearity, serving as a preparatory stage for the subsequent nonlinear transformation.

2.2. **Fractional Spline Activation Functions.** The nonlinearity is introduced via fractional causal B-spline functions applied element-wise to Z. The output of this layer is:

$$H_{i,j} = \beta_+^{\alpha_j}(Z_{i,j}), \quad i = 1, \ldots, N,\ j = 1, \ldots, m,$$

where $\beta_+^{\alpha_j}$ is the fractional causal B-spline of real order $\alpha_j > 0$, assigned to the j-th neuron. The parameters α_j are learned during training. Unlike fixed activation functions (e.g., ReLU, tanh), the shape of $\beta_+^{\alpha_j}$ varies continuously with α_j, allowing the network to adapt its local behavior: for small α_j, the function is more localized and less smooth suitable for capturing sharp transitions or fine details; for large α_j, it becomes smoother and wider promoting global regularity and numerical stability. This makes α_j a **learnable regularity parameter**, giving the model fine-grained control over the trade-off between flexibility and smoothness.

2.3. **Output Layer and Functional Reconstruction.** The final output is computed as a linear combination of the activated values:

$$Y = HC,$$

where $C \in \mathbb{R}^{m\times p}$ is the output coefficient matrix, and $Y \in \mathbb{R}^{N\times p}$ is the predicted output (for p target variables). This final layer has several important properties: the coefficients in C directly indicate the contribution of each neuron to the final prediction; the entire network can be interpreted as a weighted sum of known basis functions (the splines), enabling analytical understanding of its behavior; and each neuron corresponds to a well-defined mathematical object, enhancing model interpretability.

Backpropagation : Gradient Theorem and Proof
Consider a feedforward Fractional Spline Kolmogorov-Arnold Network with L layers, where each hidden layer $\ell = 1, \ldots, L-1$ applies fractional causal B-spline activations of real order $\alpha_j^{(\ell)} > 0$. The output layer computes the prediction as $Y = C \cdot H^{(L)}$, where $C \in \mathbb{R}^{1\times m_L}$ is the output coefficient vector, and $H^{(L)} \in \mathbb{R}^{N\times m_L}$ is the activation matrix at the last layer. The loss function is the mean squared error:

$$L = \frac{1}{N}\|Y - T\|^2,$$

where $T \in \mathbb{R}^{N\times 1}$ is the target vector, and N is the number of samples. Let $U \in \mathbb{R}^N$ denote the column vector of ones, i.e., $U = [1, 1, \ldots, 1]^\top$.
We now state and prove the main result for gradient computation in this architecture.

Theorem 2.1. *Let the network be defined with L hidden layers, output weights C, layer weights $A^{(\ell)}$, biases $b^{(\ell)}$, and order parameters $\alpha^{(\ell)}$. Let Y and T denote respectively the network output and the target, and let $H^{(\ell)}$ be the activation matrix at layer ℓ.*
For gradient evaluation, define the forward error term at the output layer by

$$F_L = \big(C^\top(Y - T)\big) \odot H^{(L)}, \tag{2.1}$$

and the backpropagated error at layer ℓ by

$$E_\ell = \big((A^{(\ell+1)})^\top E_{\ell+1}\big) \odot H^{(\ell)}, \qquad \ell = L-1, \ldots, 1. \tag{2.2}$$

Moreover, $\partial_\alpha H^{(\ell)}$ denotes the matrix whose (i,j)-th entry is given by

$$\frac{\partial}{\partial \alpha_j^{(\ell)}}, \beta_+^{\alpha_j^{(\ell)}}\big(Z_{i,j}^{(\ell)}\big). \tag{2.3}$$

The gradients of the loss function L with respect to the parameters of the output layer are given by

$$\frac{\partial L}{\partial C} = 2(Y - T)H^{(L)\top},$$
$$\frac{\partial L}{\partial A^{(L)}} = F_L H^{(L-1)\top}, \ \frac{\partial L}{\partial b^{(L)}} = F_L U^\top, \ \frac{\partial L}{\partial \alpha^{(L)}} = \Big[2(Y - T)(\partial_\alpha H^{(L)})^\top\Big] \odot C. \tag{2.4}$$

For each hidden layer $\ell = 1, \ldots, L-1$, the gradients are expressed as

$$\frac{\partial L}{\partial A^{(\ell)}} = E_\ell H^{(\ell-1)\top}, \ \frac{\partial L}{\partial b^{(\ell)}} \ = E_\ell U^\top, \ \frac{\partial L}{\partial \alpha^{(\ell)}} = \big((A^{(\ell+1)})^\top E_\ell\big) \odot (\partial_\alpha H^{(\ell)})^\top. \tag{2.5}$$

These expressions define a complete backpropagation scheme for all trainable parameters of the proposed network.

Proof. The proof proceeds in three stages, corresponding to the backward propagation of errors through the network.

Step 1: Gradient with Respect to Output Coefficients C. By direct differentiation,

$$\frac{\partial L}{\partial C} = 2(Y - T) \cdot H^{(L)^\top}.$$

Since the factor $1/N$ is typically absorbed into the learning rate during optimization, we write:

$$\frac{\partial L}{\partial C} = 2(Y - T) \cdot H^{(L)^\top}.$$

This completes the first part of the theorem.

Step 2: Definition of Error Vectors. Define the error vector at the output layer as:

$$F_L = \left(C^\top \cdot (Y - T)\right) \odot H^{(L)}.$$

This vector captures the contribution of each neuron's activation to the final error, weighted by its output coefficient and modulated by its current value.
For hidden layers $\ell = L - 1, \ldots, 1$, define recursively:

$$E_\ell = \left[\left(A^{(\ell+1)}\right)^\top \cdot E_{\ell+1}\right] \odot H^{(\ell)}.$$

This recurrence propagates the error backward from layer $\ell + 1$ to layer ℓ, incorporating the local derivative of the activation function via the Hadamard product with $H^{(\ell)}$.

Step 3: Gradients with Respect to Weights, Biases, and Fractional Orders.
Gradients w.r.t. Weights and Biases.
The gradient of L with respect to the weights of the last layer is obtained by the chain rule:

$$\frac{\partial L}{\partial A^{(L)}} = \frac{\partial L}{\partial H^{(L)}} \cdot \frac{\partial H^{(L)}}{\partial A^{(L)}} = F_L \cdot \left(H^{(L-1)}\right)^\top.$$

The gradient with respect to the bias vector is:

$$\frac{\partial L}{\partial b^{(L)}} = F_L \cdot U^\top,$$

since adding $b^{(L)}U^\top$ to the preactivation introduces a constant shift across all samples, and the gradient sums over them.
For hidden layers $\ell = 1, \ldots, L - 1$, the gradients are:

$$\frac{\partial L}{\partial A^{(\ell)}} = E_\ell \cdot \left(H^{(\ell-1)}\right)^\top, \quad \frac{\partial L}{\partial b^{(\ell)}} = E_\ell \cdot U^\top.$$

These follow directly from the definition of E_ℓ and the structure of the affine transformation $Z^{(\ell)} = A^{(\ell)}H^{(\ell-1)} + b^{(\ell)}U^\top$.

Gradients w.r.t. Fractional Orders $\alpha^{(\ell)}$.
The key property is that the activation functions are differentiable with respect to their fractional order. Let $\partial_\alpha H^{(\ell)}$ denote the matrix of partial derivatives of the activations with respect to $\alpha^{(\ell)}$, computed element wise.
At the output layer, the gradient is:

$$\frac{\partial L}{\partial \alpha^{(L)}} = \left[2(Y - T) \cdot \left(\partial_\alpha H^{(L)}\right)^\top\right] \odot C$$

For hidden layers $\ell = 1, \ldots, L-1$, using the error vector E_ℓ, we have:

$$\frac{\partial L}{\partial \alpha^{(\ell)}} = \left[(A^{(l+1)})^\top \cdot E_\ell\right] \odot \left(\partial_\alpha H^{(\ell)}\right)^\top$$

This follows from the chain rule: the sensitivity of the loss to $\alpha^{(\ell)}$ is mediated through the error propagated to layer ℓ, which is then multiplied by the local derivative of the activation with respect to $\alpha^{(\ell)}$.
All expressions in the theorem are thus derived. This completes the proof. □

2.4. **Parameter Update: A Differentiated Approach for Constrained Parameters.**

All model parameters including weights, biases, output coefficients, and fractional orders undergo optimization via gradient based methods. However, the update mechanisms for these parameters are not uniform. While the majority of parameters evolve according to standard gradient descent procedures, the fractional orders α_j necessitate distinct treatment due to inherent mathematical constraints. The activation functions employed in this model are defined as fractional causal B-splines $\beta_+^{\alpha_j}(x)$, which require the parameter α_j to remain strictly positive. Should α_j assume a non positive value, the associated spline may become mathematically undefined, numerically unstable, or lose its desired smoothness properties, thereby compromising the structural integrity of the model. Consequently, unlike other parameters that can assume any real value, α_j must be maintained within the domain $\alpha_j > 0$ throughout the entire training process. This constraint is not merely a practical consideration but is fundamentally embedded within the mathematical formulation of the model [17].
To ensure adherence to this constraint without disrupting the optimization dynamics, we implement Projected Gradient Descent for updating the α_j values. The procedure unfolds as follows. First, the gradient of the loss function L with respect to α_j is computed identically to gradients for unconstrained parameters. Second, an intermediate update is performed by stepping in the direction of steepest descent:

$$\alpha_j^{\text{temp}} = \alpha_j - \eta \cdot \frac{\partial L}{\partial \alpha_j}.$$

Third, the resulting value is projected onto the feasible set $\alpha_j > 0$ through the operation:

$$\alpha_j \leftarrow \max(\alpha_j^{\text{temp}}, \epsilon),$$

where $\epsilon = 10^{-6}$ serves as a small positive constant introduced to mitigate numerical instabilities near zero. This projection mechanism functions as a safeguard; should the gradient driven update attempt to drive α_j below zero, the parameter is gently adjusted back into the permissible region. The learning process continues uninterrupted while preserving the mathematical validity of the model.
For all other parameters namely the weights $A^{(\ell)}$, biases $b^{(\ell)}$, and output coefficients C no such constraints apply. These parameters are updated using conventional gradient descent:

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta),$$

without modification. Their unrestricted evolution across the real number line enables the network to adapt flexibly to the underlying data structure.
This differentiated strategy yields several critical advantages. It ensures the mathematical integrity of the splines at every iteration, maintains numerical stability by avoiding divisions by zero or evaluations of undefined Gamma functions, and preserves end to end differentiability. The projection operation is sufficiently straightforward to maintain gradient flow through the remainder of the network. In essence, the model is permitted to learn freely while the α_j parameters are guided to remain within their natural boundaries. This equilibrium between flexibility and constraint underpins the robustness and efficacy of of this model in scientific applications [19].

2.5. **Experimental evaluation.**

This section evaluates the numerical performance of the proposed network which combines Kolmogorov-Arnold architecture with learnable fractional spline activations on several one-dimensional regression problems with contrasting regularity properties. The selected target functions illustrate three common situations: rapid oscillations, discontinuities, and localized smooth structures.
The model is compared to a standard multilayer perceptron (MLP) of equivalent size. All experiments were performed using `NumPy` for numerical calculations, `scipy` for some special functions, and `matplotlib` for visualization, with identical hyperparameters to ensure a fair comparison.

Oscillating function :
The target function is :

$$y(x) = \tfrac{1}{2}\cos(x) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 0.1^2), \quad x \in [0, 5],$$

The target function is sampled in 200 uniformly distributed points.

TABLE 1. Quantitative comparison of models on the Noisy Cosine function

Function	**New NN (MSE)**	**MLP (MSE)**	**Poly (MSE)**	**Advantage (%)**
Noisy Cosine	0.008324	0.009549	0.010118	17.74

Smooth localized function :
The third experiment concerns a smooth function with localized support:

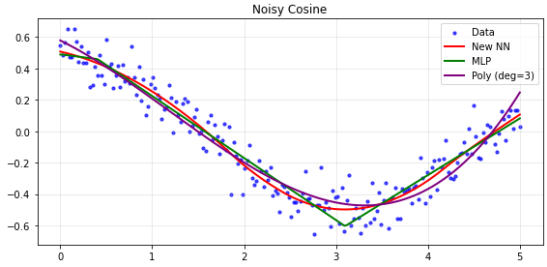

FIGURE 3. Comparison of New NN, MLP, and Polynomial Regression on Noisy Function Approximation.

$$f(x) = \exp\Big(-\tfrac{(x-2.5)^2}{2\cdot 0.8^2}\Big) + \varepsilon(x), \quad \varepsilon(x) \sim \mathcal{N}(0, 0.05^2), \quad x \in [0,5].$$

This model reaches a significantly lower error, with a relative gain of 60.83%.

TABLE 2. Quantitative Comparison on Noisy Gaussian Function

Function	**New NN (MSE)**	**MLP (MSE)**	**Poly (MSE)**	**Advantage (%)**
Noisy Gaussian	0.002178	0.005560	0.028084	92.25

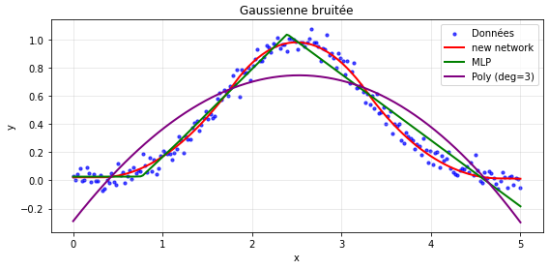

FIGURE 4. Approximation of the noisy Gaussian function. with This model and MLP.

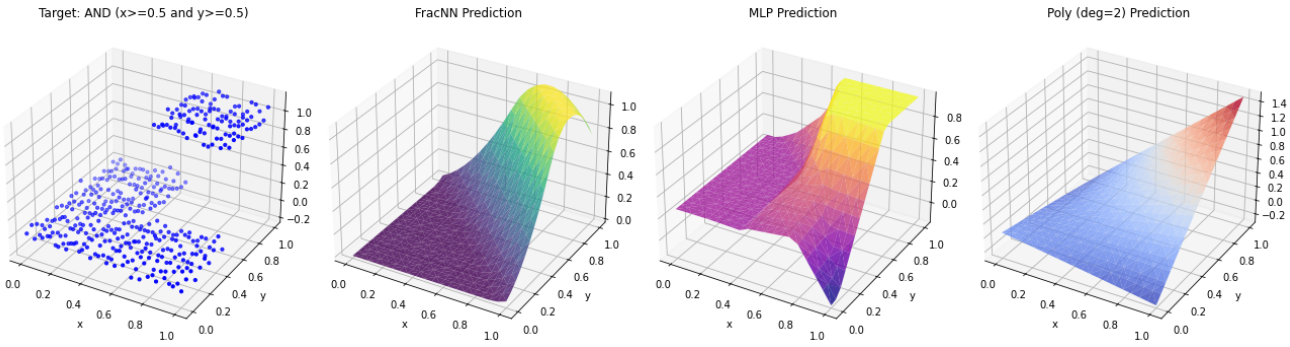

FIGURE 5. Comparison of FracNN, MLP, and Polynomial Regression on the 2D Logical AND function

Explanation. The target function is a 2D logical AND: output = 1 if both inputs are ≥ 0.5, else 0. This creates a sharp discontinuity in the top-right corner of the unit square $[0, 1] \times [0, 1]$. We add small Gaussian noise to simulate real-world data.

The FracNN model (center) successfully learns the sharp boundary, producing a prediction surface that closely matches the true step-like structure. In contrast, the MLP (right) produces a smooth, curved surface that fails to capture the sharp transition a known limitation of ReLU-based networks on discontinuous functions. The polynomial regression (not shown here, but quantified below) performs even worse, as low-degree polynomials cannot represent such abrupt changes.

Quantitatively, FracNN achieves the lowest mean squared error (MSE = 0.060), outperforming both MLP (MSE = 0.340) and polynomial regression (MSE = 0.165). This represents a 63% relative improvement over polynomial regression, demonstrating the advantage of adaptive fractional splines in modeling non-smooth, logic-based functions.

This experiment confirms that FracNN is better suited than standard MLPs or polynomial models for tasks requiring precise decision boundaries, such as classification or control systems.

Summary of results :

The results show that this network consistently outperforms the MLP across all tested functions. This is because it can adapt its local smoothness through the learned α parameters. It adjusts to the shape of the data capturing sharp jumps, fast oscillations, or localized peaks leading to better accuracy.

3. Conclusion

In this work we introduced FS KAN a new kind of neural network that brings together two powerful ideas the structure of Kolmogorov Arnold networks and the flexibility of fractional splines. What makes it special It does not just learn weights and biases it also learns how smooth each connection should be through a parameter called $\alpha_j > 0$. This lets the model adjust itself to the data's shape. Our experiences show that this adaptability is effective. In all tested scenarios, from noisy functions to localized discontinuities and spikes, this model consistently outperforms a classic MLP. Moreover, unlike many deep models, each component remains interpretable, which allows understanding the role of each neuron in the approximation.

The math behind it is solid. We built on the idea that fractional splines can change their behavior smoothly as α changes and we showed how to compute gradients for α during training using projected gradient descent to keep everything stable.

This is only a first step. Our experiments, limited to simple one dimensional problems and single layer architectures, nevertheless show that the principle works: allowing the model to adjust its own regularity leads to better performance and provides a clearer understanding of how it works. Real data, which is more complex and noisy, remains to be explored, but this approach paves the way for adaptive and interpretable models that can automatically adjust to the local

structures of the functions they learn. In fields such as medical imaging, and financial modeling, where accuracy is crucial but understanding the results is equally important, this model proves particularly well suited. In science, the best models do not just predict: they also explain.

References

1. Abbas, M., Aslam, S., Abdullah, F. A., Riaz, M. B., & Gepreel, K. A. (2023). An efficient spline technique for solving time-fractional integro-differential equations. *Heliyon*, *9*(9), e19307. https://doi.org/10.1016/j.heliyon.2023.e19307
2. Arnold, V. I. (1957). On functions of three variables. *Doklady Akademii Nauk SSSR*, *114*, 679–681.
3. Blu, T., & Unser, M. (2000). Fractional splines and wavelets. *SIAM Review*, *42*(1), 43–67. https://doi.org/10.1137/S0036144598349435
4. Chen, Y. Q., Sun, R., & Zhou, A. (2007). An overview of fractional order signal processing (FOSP) techniques. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 4806, pp. 1205–1222). https://doi.org/10.1115/DETC2007-34228
5. Ciesielski, M. (2024). Numerical algorithms for approximation of fractional integrals and derivatives based on quintic spline interpolation. *Symmetry*, *16*(2), 252. https://doi.org/10.3390/sym16020252
6. Féjoz, J. (2004). Démonstration du théorème d'Arnold sur la stabilité du système planétaire (d'après Herman). *Ergodic Theory and Dynamical Systems*, *24*(5), 1521–1582.
7. Forster, B., & Massopust, P. (2011). Multivariate interpolation with fundamental splines of fractional order. *PAMM*, *11*(1), 857–858. https://doi.org/10.1002/pamm.201110416
8. Ji, T., Hou, Y., & Zhang, D. (2024). *A comprehensive survey on Kolmogorov-Arnold networks (KAN)*. arXiv preprint arXiv:2407.11075. https://arxiv.org/abs/2407.11075
9. Khlefha, A. (2024). Comprehensive review of interpolation techniques using B-spline. *International Academic Journal of Science and Engineering*, *11*(1), 304–311.
10. Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR*, *114*(5), 953–956.
11. Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., & Tegmark, M. (2024). *KAN: Kolmogorov-Arnold networks*. arXiv preprint. https://doi.org/10.48550/arXiv.2404.19756
12. Massopust, P. (2014). Fractional operators, Dirichlet averages, and splines. In P. L. Butzer, J. S. P. Bhatia, & R. L. Stens (Eds.), *New perspectives on approximation and sampling theory: Festschrift in honor of Paul Butzer's 85th birthday* (pp. 399–422). Springer.
13. Meijering, E. H. W. (2000). Spline interpolation in medical imaging: Comparison with other convolution-based approaches. In *Proceedings of the 10th European Signal Processing Conference* (pp. 1–8). IEEE.
14. Oumellal, F., Mouhoub, C., Lamnii, M., & Lamnii, A. (2025). A new quadrilateral finite element based on quadratic spline. *Gulf Journal of Mathematics*, *21*(2), 259–274. https://doi.org/10.56947/gjom.v21i2.3622
15. Pinto, M. G. de F. (2024).Analyzing and modeling long-memory time series using fractional spline wavelets (Doctoral dissertation). Universidade de São Paulo.
16. Pinto, M. G. de F., & Chiann, C. (2024). A maximum-likelihood-based approach to estimate the long memory parameter using fractional spline wavelets. *Signal Processing*, *222*, 109518. https://doi.org/10.1016/j.sigpro.2024.109518
17. Unser, M., & Blu, T. (2000). Fractional splines and wavelets. *SIAM Review*, *42*(1), 43–67. https://doi.org/10.1137/S0036144598349435
18. Unser, M. (2002). Sampling—50 years after Shannon. *Proceedings of the IEEE*, *88*(4), 569–587. https://doi.org/10.1109/5.929663

19. Valarmathi, R., & Gowrisankar, A. (2023). On the variable order fractional calculus of fractal interpolation functions. *Fractional Calculus and Applied Analysis*, *26*(3), 1273–1293. https://doi.org/10.1007/s13540-023-00148-8
20. Xu, Z. (2009). Multivariate F-splines and fractional box splines. *Journal of Fourier Analysis and Applications*, *15*(5), 723–738. https://doi.org/10.1007/s00041-009-9076-4

[1] LANO Laboratory, Department of Mathematics, Faculty of Sciences, Mohammed First University, Oujda, Morocco.
Email address: amal.bouaicha@ump.ac.ma
Email address: m_lamnii1@yahoo.fr
Email address: soumia.ngadi.d24@ump.ac.ma