www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(1), Jan-April 2026, 117-131. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2026.v17i1.1244

Towards Interpretable Inverse Model Control: The Role of Gray-Box Models

Marco A. Márquez-Vera[1], Alfian Ma'arif[2], Ocotlán Díaz-Parra[1], Zaineb Yakoub[3], Julio C. Ramos-Fernández[1]
[1] Department of Mechatronics Engineering, Polytechnic University of Pachuca, Mexico
[2] Department of Electrical Engineering, Universitas Ahmad Dahlan, Indonesia
[3] Department of Electrical Engineering, University of Gabès, Tunisia
Emails: marquez@upp.edu.mx, alfianmaarif@ee.uad.ac.id, ocotlan_diaz@upp.edu.mx, yakoubzaineb@yahoo.fr, jramos@upp.edu.mx

Abstract. This work presents and compares two approaches for inverse modelling and control of a biotechnological system: a fuzzy rule-based model and a Kolmogorov–Arnold network (KAN) model. Both models aim to generate a control law that enables the system to reach a desired substrate concentration by inverting the model. While the fuzzy model provides interpretability through linguistic rules, the KAN model offers an explicit functional representation that permits analysis and visualisation of each input's contribution through univariate functions. It is shown how the models can be inverted online to determine the required dilution rate, producing controlled trajectories close to the reference. The results confirm that the use of interpretable models is not only feasible for control tasks, but also provides transparency, pruning capability, and automatic generation of symbolic expressions, which support the determination and adaptation of their implementation in critical systems.
Keywords: Gray Boxes; Kolmogorov-Arnold Networks; Inverse Model; Fuzzy Logic.

Article Info
Received Sep 14, 2025
Accepted November 14, 2025

1 Introduction

There are various applications of artificial intelligence, which is a metaphor of what happens in the natural world. It involves imitating something that occurs in nature to make a decision or solve a problem. For this purpose, algorithms are created that are commonly called artificial intelligence (AI). Although, there are disagreements with this term and soft computing is also often used as a synonym, currently soft computing is part of AI and tends to focus more on fuzzy logic (Konar, 2000). For something to be classified as intelligent, it must meet some characteristics. After all, kicking a ball is not playing football, just as processing text does not know about semantics (Searle, 1990). Although, currently it is not easy to refute that the Turing test has been passed by modern artificial intelligence applications (Wang et al., 2024).

Some of the approaches taken from nature to implement algorithms capable of solving problems, optimizing results, or making decisions are, for example, genetic algorithms (GA) that simulate the survival of the fittest, where each individual represents a solution and the fittest cross to have a new generation that may have better performance. To avoid falling into a local minimum, mutations are also simulated in new generations (Surma et al., 2025). In the case of using GA for control applications, it is possible to propose that each individual represents a PID controller, with each of its gains encoded in the individual's genes. Thus, when evaluating some performance criterion that considers settling time, overshoot, steady-state error, among some other characteristic, a good option that satisfies the requirements can be found after simulating several generations (Hong et al., 2023).

GA provide the best solution found, but do not provide a controller based on soft computing, generally other AI techniques are used to propose nonlinear controllers, one of them are artificial neural networks (ANN), which mimic the connections between biological neurons to generate signals, make decisions, or classify information. ANN usually require the desired behavior (supervised learning) to adjust their parameters to achieve the expected result, this stage is called training. In addition to the network parameters to be implemented, there are also hyperparameters that are chosen arbitrarily; at this stage, GA as well as other biomimetic or bioinspired algorithms are an alternative to determine the hyperparameters to be used in the network (Khosravi and Bahram, 2025).

A neural controller is capable of adjusting itself to control a nonlinear system, having its power in the hidden layer of neurons. Given the complexity of adjusting the network parameters, it is not possible to propose the synaptic weights analytically, so training methods such as gradient descent are required. For example, Ammara et al. (2024) implemented a sliding mode control for power converters, whose reference and management are determined by an ANN to obtain maximum power from a photoelectrochemical cell. These neural controllers function as black boxes because their behavior cannot be understood until they are simulated or evaluated; consequently, modifying any parameter or hyperparameter in the ANN produces unpredictable outputs.

Although ANN have many applications and are used not only for control purposes, there is also fuzzy logic that can be designed based on expert knowledge, or even, if some adaptation technique of the fuzzy system is used, the resulting system can be interpreted and have an idea or approximation of what would happen if some parameter is modified. This way, a person's experience or an expert's knowledge can be used to propose the fuzzy system or controller (Bárcenas-Castañeda et al., 2023). Thus, fuzzy logic can be interpreted as a gray box, where perhaps there is no analytical expression of its behavior, but there is an idea of what it will deliver according to its configuration.

This is because it is a tool designed to imitate the way we make decisions from scarce or ambiguous information, thus achieving a degree of interpretability (Ouifak and Idri, 2025).

A fuzzy control can be proposed based on normalizing its inputs or by knowing the range of values that each of the inputs can have, a simple example would be to propose the fuzzy rules for a PD-type control, for example a negative error can represent that the output is above the reference, then the control signal should decrease, or if the derivative of the error is positive, it means that the error is becoming large. Although, it neither is nor know if the output is moving away above or below the reference, thus by combining these two parts a fuzzy PD control can be proposed. The number of membership functions used for each input will give us how smooth the change in the control signal would be when calling, for example, the error with labels such as very negative, negative, zero, or positive. This way, linguistic variables are worked with instead of numerical variables. An example of this is the work presented by Nguyen et al. (2025), who propose a table to represent the fuzzy rules based on their decisions, having in the secondary diagonal the fuzzy set that represents zero in the output signal, saturating in the upper left corner with the most positive value of the control signal, this is for the case where the inputs (error and error derivative) are represented by the fuzzy sets called "negative big", and the lower right corner has as output the most negative fuzzy set of the control signal.

If these two techniques capable of making a nonlinear controller are combined, neuro-fuzzy controllers are obtained, where the capabilities of approximating signals and generalizing results of ANN are complemented with the capabilities of using expert knowledge, and handling ambiguous information of fuzzy logic. This type of system can be interpreted as an ANN whose neurons are fuzzy sets or operators, or as a fuzzy system that is trained as if it were an ANN. An interpretation or approximations of these neuro-fuzzy systems are radial basis functions (RBF). RBF are neural networks that have a Gaussian bell as activation function instead of a sigmoid or a hyperbolic tangent, similarly it can be seen as a fuzzy system where the sets used to make the fuzzy partition of a variable are done by Gaussian bells (Kassem and Çamur, 2017).

It should be noted that ANN do not handle signals similar to biological neurons, since they generally use the sigmoid and hyperbolic tangent as activation functions, which give a value between zero and one, or between minus one and one respectively. In the case of deep ANN (deep learning), it is common to use the rectified linear unit (ReLU) as activation function, which keeps any negative evaluation by the neuron at zero. In nature, neurons handle voltage potentials in the range of millivolts and produce voltage spikes if activated. Spiking neural networks are capable of mimicking the signals generated by biological neurons, but they remain black boxes, and it is necessary to encode the signals delivered by this type of network. There is encoding by the instant when the voltage spike occurs before the potential in the neuron resets, and the average of spikes obtained within a time window. An interesting application was developed by Wang (2015), who proposed a tool to simulate cortical signals and their interface with spiking neural networks.

ANN are based on the universal approximation theorem that mentions the existence of a neural network, if the ANN has at least one hidden layer with a sufficient number of neurons, and with a nonlinear activation function, is capable of approximating any continuous function on a bounded interval, and with arbitrary accuracy. Although, a deep ANN has the problem of gradient vanishing, by increasing the number of neurons and iterations or training epochs, the approximation error can be reduced. A similar theorem is the Kolmogorov representation theorem, which states that a multivariate continuous function $f(x_1, x_2, ..., x_n)$ can be represented exactly as a finite composition of continuous functions of one variable, and their sums. Thus, the

approximation can be exact and not only approximated with arbitrary accuracy. But it can be difficult to find the functions of one variable that would do it (Thakolkaran et al., 2025). Based on Kolmogorov's theorem, a multivariate function $f(x)$ can be decomposed as:

$$f(x) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right), \tag{1}$$

where Φ_q and $\varphi_{q,p}$ are continuous functions of one variable.

Based on the previous theorem, Kolmogorov-Arnold networks (KAN) have recently emerged where instead of having synaptic weights between neurons, those "weights" are functions that can be learned, and instead of an activation function in the soma of the neuron, there is the sum of the learned functions that arrive through the neuron's connections. These KAN can present better results than ANN using even fewer neurons (now called nodes), in addition to the fact that both the functions learned in the connections or axes and the output of each node can be graphed and even visualized by a user, making this type of network interpretable (gray boxes). Furthermore, Liu et al. (2024) mentioned that KAN can be helpful in finding or rediscovering laws in mathematics and physics. In their work they use Kolmogorov's theorem to propose a new structure, similar to ANN, where the univariate functions are polynomials, specifically B-spline cubic splines, and once calculated, the resulting functions can give an idea of the composite functions that describe the system or phenomenon to be described.

Additionally, KAN can be made multilayer, allowing finding composite functions of the inputs. For example, if the function to be approximated by a KAN is $f(x_1, x_2) = \sin x_1^2 + \log x_2$, the polynomials found will be very similar to these functions within the domain with which they are trained, so a network could have two layers where the axis (connection) from x_1 to the next node would be described by a polynomial whose shape is very similar to a parabola (x^2), while for the second input, the obtained curve would be the shape of the function $\log(x_2)$, from this node another connection goes to a second layer, whose polynomial or cubic spline gives the shape of a sinusoidal (see Fig. 1). Initially, it is not simple to propose the number of axes or nodes to build the KAN, but pruning can be implemented to discard the axes that hardly contribute to the nodes they connect to and thus simplify the network.

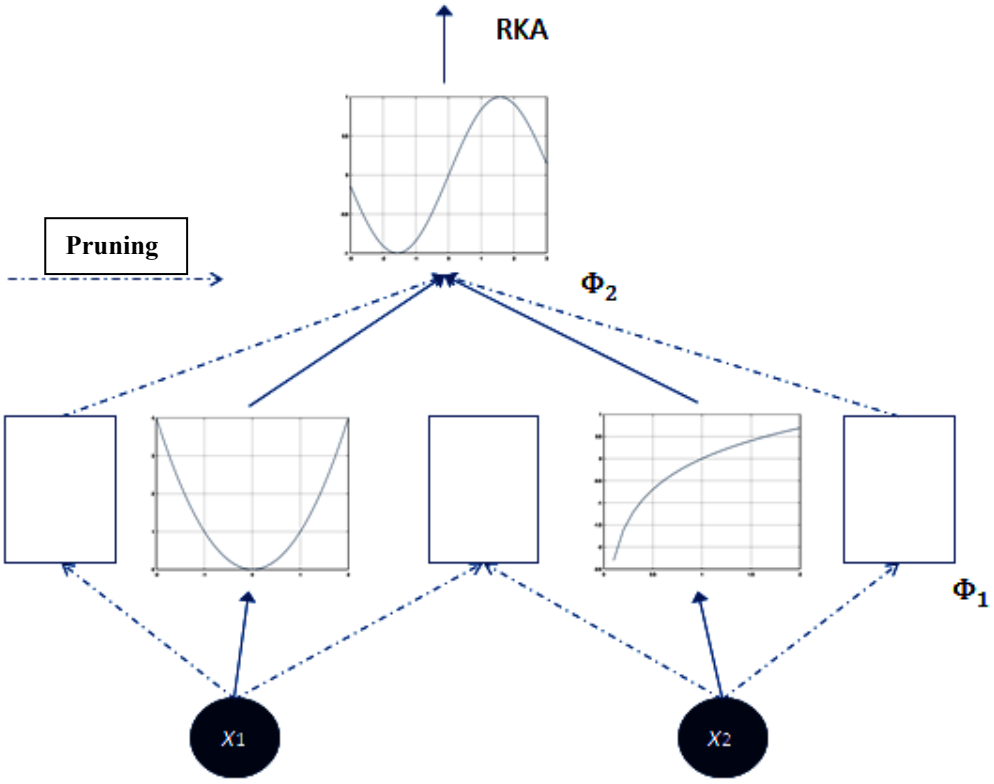

Fig. 1. Example of a KAN for approximating a function.

Given the capacity of having interpretability of some AI models, this article proposes to use fuzzy logic and KANs to model a dynamic system to subsequently invert the models in order to control the system. Thanks to the interpretability of gray boxes, it is sought to describe both the system model and its control through rules involving linguistic variables (fuzzy logic), and through the polynomials given by KAN.

A general scheme of how to do this type of control is shown in Fig. 2, where the model is calculated to predict the next system output from the known information, and then the control signal that would cause the desired output is calculated by inverting the model that must have some degree of interpretability.

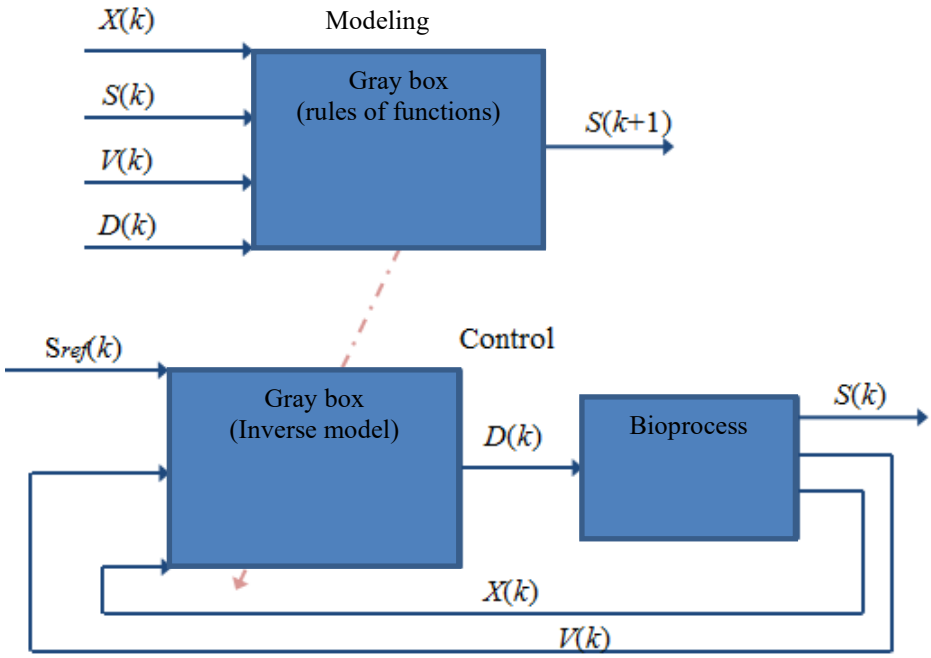

Fig. 2. Scheme of gray box model inversion.

The rest of the article is organized as follows: Section 2 describes how to obtain a fuzzy model, and the way to invert it to have the desired behavior of the system to be controlled. Section 3 describes the use and inversion of a KAN to control the same system; in this case a biotechnological process is controlled. Section 4 contains some comments on the interpretability of the obtained gray boxes and gives some points on their application and improvements. Finally, Section 5 presents the conclusions.

2 Modeling and Control using Fuzzy Logic

Fuzzy logic control offers a suitable approach for managing complex bioprocesses due to its inherent ability to handle nonlinear systems without requiring precise mathematical models. Unlike conventional control techniques, fuzzy logic can effectively accommodate system uncertainties and non-minimum phase behaviors through its rule-based framework. This capability is especially valuable for the system under consideration which is a bioprocess characterized by nonlinear dynamics. These bioprocesses exhibit non-minimum phase behavior, resulting in an inverse response. Linearization of the mathematical model yields a transfer function with unstable zeros (Márquez-Vera et al., 2022). The specific bioprocess addressed involves biodegradation, where maintaining a constant reference is essential for the continuous biodegradation of xenobiotic compounds.

The mathematical model of the simulated bioprocess is given by:

$$\dot{X}(t) = \frac{\mu S(t) X(t)}{k_S + S(t)} - D(t)X(t),$$
$$\dot{S}(t) = -k\frac{\mu S(t) X(t)}{k_S + S(t)} - D(t)(S(t) - S_{in}),$$
$$\dot{V}(t) = D(t)V(t), \tag{2}$$

where $X(t)$ is the biomass concentration inside the reactor and it is assumed that due to agitation inside the reactor, the concentrations are equal at any point inside the reactor. $S(t)$ is the substrate concentration (microorganisms' food). $V(t)$ is the volume of liquid inside the reactor. S_{in} is the input substrate concentration, which for control purposes is usually kept constant, and in this research it is 5 mg/l. The input to the system is usually taken as the dilution rate $D(t)$ which is given by the ratio between the input flow and the volume of liquid inside the reactor. μ represents the specific growth rate, being 2/min. k_s is the saturation constant, and has a value of 4 mg/l. k is the kinetic constant and being negative in (2), means that the substrate is consumed, the value of this constant is 0.9/min.

To obtain a fuzzy model, a fuzzy partition of each variable involved in the model is required, to guarantee a monotonic model, two membership functions determined with the maximum and minimum values obtained in the simulation are used, where the input signal varies in time and is defined by $D(t) = 0.1\ \sin(0.2t) + 0.06\ \sin(0.8t) + 0.06$. In this case, when performing the simulation, it was obtained that the values, for example of the substrate concentration $S(t)$ is between 0.2 and 1.6 mg/l. For this variable the membership functions used for its fuzzy partition are shown in Figure 3. The other ranges used in this work are: For biomass concentration [0, 2.6 mg/l], volume between 1 and 2.2 l, for the dilution rate the range is [-0.05, 0.18 1/l].

There were used only two membership functions to make the fuzzy partition of variables to guarantee the monotonicity of the resulting model for its inversion.

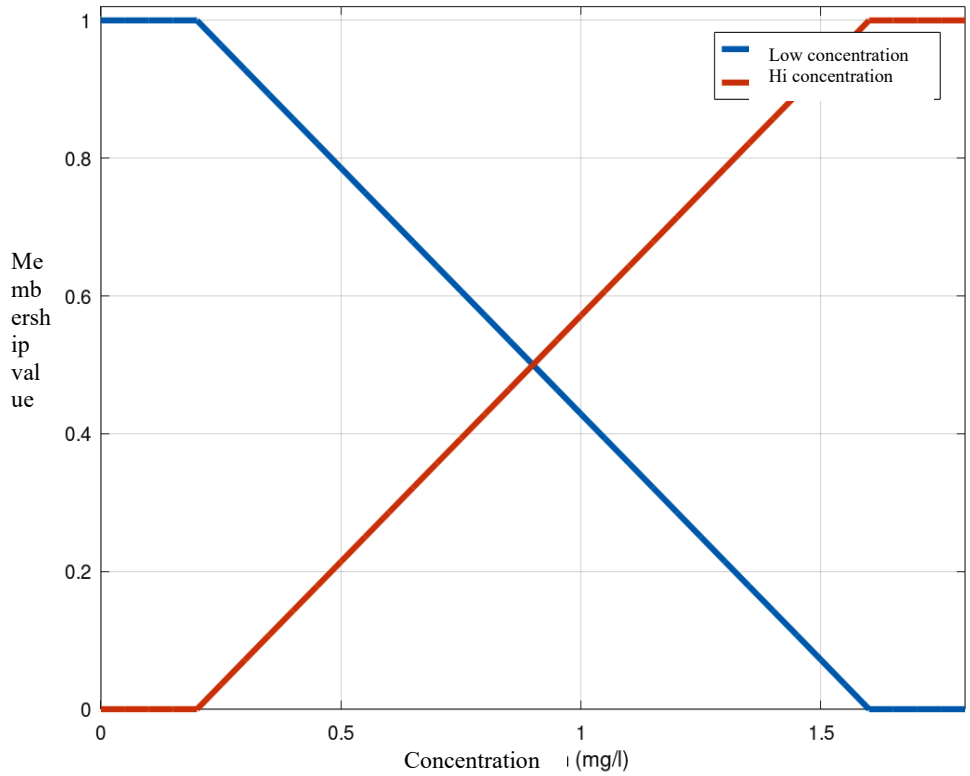

Fig. 3. Membership functions of the fuzzy partition of the substrate concentration.

2.1 Fuzzy Modeling

Once the membership functions are proposed to evaluate the mathematical model variables $X(t)$, $S(t)$, $V(t)$ and $D(t)$, a matrix is generated that contains the membership values at each time instant for each variable, this matrix will have the dimension $\beta \in R^{n\times 2^4}$ because there are four variables and n is the number of samples per variable. Once this matrix is filled with the memberships, it is normalized by dividing each column by the sum of the memberships of that column, thus obtaining:

$$\Gamma_{i,j} = \frac{\beta_{i,j}}{\sum_{k=1}^{n} \beta_{i,k}}, \tag{3}$$

The next thing is to find the consequents of the fuzzy rules, the antecedents are formed by the evaluation of the fuzzy sets of each input variable, being in this case $2^4 = 16$ fuzzy rules. The consequents are scalars, thus having a Takagi-Sugeno type inference system (Aslam and Bilal, 2024), and these consequents are obtained by:

$$B = (\Gamma^T \Gamma)^{-1} \Gamma^T S_{+1}, \tag{4}$$

where S_{+1} represents the values of the substrate concentration but advanced one sample.

$$S_{fuzzy}(i) = \frac{\sum_{k=1}^{n} \beta_{i,k} B_k}{\sum_{k=1}^{n} \beta_{i,k}}, \tag{5}$$

Thus, the fuzzy approximation of the substrate concentration, denoted as S_{fuzzy} is the evaluation of the fuzzy rules, where β represents the membership values (antecedents), and B are the consequents of the rules. This type of modeling uses sector nonlinearity (Tanaka and Wang, 2001).

Figure 4 shows the simulation of the bioprocess with the input dilution rate $D(t)$ mentioned above, and the response of the found fuzzy model. The consequents B are used to perform the inversion of the model, since in fuzzy modeling, all variables are inputs to the fuzzy system ($X(t)$, $S(t)$, $V(t)$, and $D(t)$) and the idea is to approximate the next sample of the substrate concentration $S_{+1}(t)$. Now the inverse model will deliver the dilution rate $D(t)$ that would provide the reference concentration $S_{ref}(t)$.

A limitation of this technique is to have a monotonic model. Generally, if at least two variables are involved in the model, the use of two membership function will produce an invertible fuzzy model, but if to approach the nonlinearities in the model make necessary to use more membership functions for the fuzzy partition of variables, then the rule-consequents must be evaluated in order that for certain input, the consequent always grows up, or it decreases in all evaluations.

2.2 Inverse Fuzzy Model Control

If it obtained an acceptable fuzzy model, the consequents of the rules must be organized in the matrix C so that the first column contains the consequents related to the first fuzzy set of the dilution rate (linguistic variable "Low dilution"), the reorganized values of B are shown in Table 1, and are the values contained in C.

The next step is to evaluate through the fuzzy sets the variables $X(t)$, $S(t)$ and $V(t)$ to now evaluate $2^3 = 8$ fuzzy antecedents that are aggregated in the vector h, which will be used to determine the ranges with which to build two new fuzzy sets that will describe the memberships of the reference $S_{ref}(t)$, these sets are created online as the system evolves. Thus, the range limits Λ for the fuzzy sets that will describe the reference are obtained using:

$$\Lambda_j = \sum_{k=1}^{2} \lambda_k C_{k,j}, \tag{6}$$

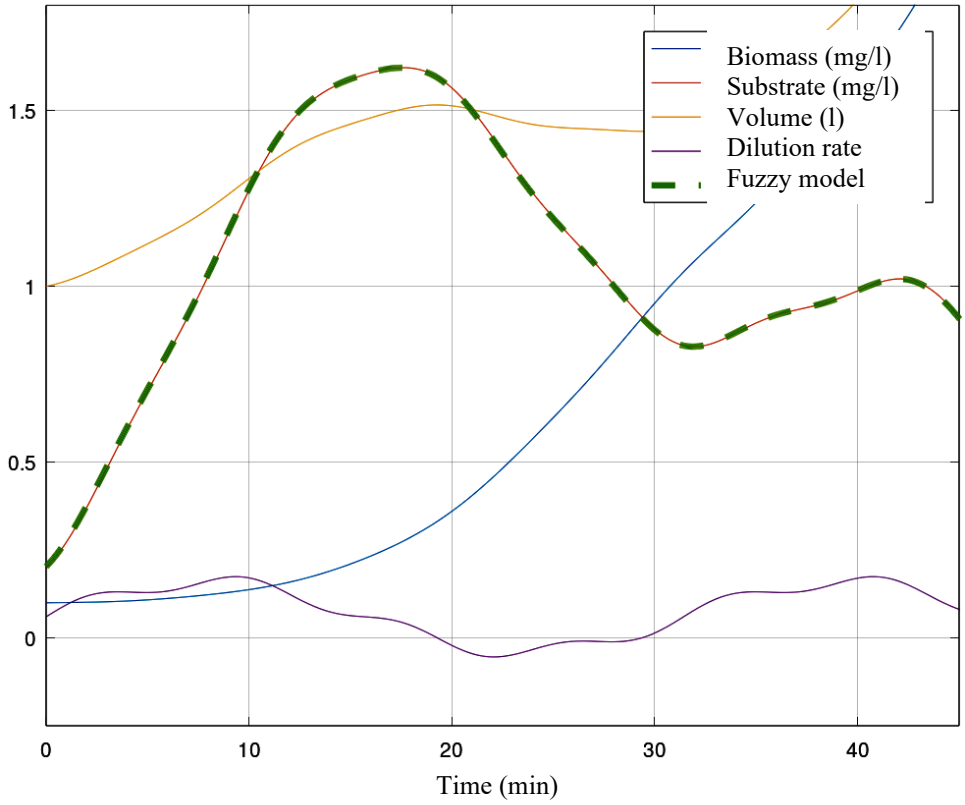

Fig. 4. Simulation of the bioprocess and result of the fuzzy model of the substrate.

Now, the calculation of the control signal $D(t)$ to have the desired value $S_{ref}(t)$ is obtained by evaluating the reference with the fuzzy sets obtained with the values Λ, denoting these two memberships as $\mu_{\Lambda 1}$ and $\mu_{\Lambda 2}$, the dilution rate to use is:

$$D = (\mu_{\Lambda_1} \quad \mu_{\Lambda_2})(-0.05 \quad 0.18)^T, \tag{7}$$

By applying this dilution rate D calculated from building the fuzzy sets to evaluate the reference (Márquez-Vera et al., 2016), the result shown in Fig. 5 is obtained, where it is possible to see the calculated dilution rate. It is also possible to note the inverse response of the system, and how close the substrate concentration $S(t)$ is to the reference.

Table 1. *B* consequents reordered

Low dilution	High dilution
0.25008	0.27789
3.50003	4.78306
0.85791	0.58297
3.16506	3.76554
-1.0974	-1.5381
0.62897	0.69311
-0.4776	-1.2519
0.23178	-0.2277

3 Modeling and Control using Kolmogorov-Arnold Networks

The fuzzy logic gives a model in rules that need to be interpreted by the operator; if an easy mathematical description is required the KAN is an alternative. To perform a model using KAN, the network needs to be initialized to train it based on the simulated data, in this case the same dilution rate used to build the fuzzy model will be used. KAN can be made deep to improve their response and better fit the data, running the risk of overgeneralization. In this work, it is proposed to use two hidden layers to avoid overfitting, as well as to maintain the interpretability of the network. This network will have nodes as neurons, and instead of simple synaptic connections there are polynomials, these are univariate functions and the connections are called axes. When these polynomials arrive, weights are calculated to determine the proportion in which they influence the node they connect to.

The polynomials used are B-splines, typically cubic polynomials. The degree of the polynomial k (order k+1) is defined over instants t_0, t_1, ..., t_m (equally spaced in this article). Thus the spline **B** is defined as:

$$\mathbf{B}_{i,k}(x) = \begin{cases} 1, \text{if } t_i \leq x < t_{i+1}, \text{and } k = 0, \\ 0, \text{other case}, \end{cases} \tag{8}$$

and for a k>0, **B** will be:

$$\mathbf{B}_{i,k}(x) = \alpha_{i,k}\mathbf{B}_{i,k-1} + (1 - \alpha_{i+1,k})\mathbf{B}_{i+1,k-1}, \tag{9}$$

where

$$\alpha_{i,k} = \begin{cases} \dfrac{x - t_i}{t_{i+k} - t_i}, \text{if } t_{i+k} \neq t_i, \\ 0, \text{other case}. \end{cases}$$

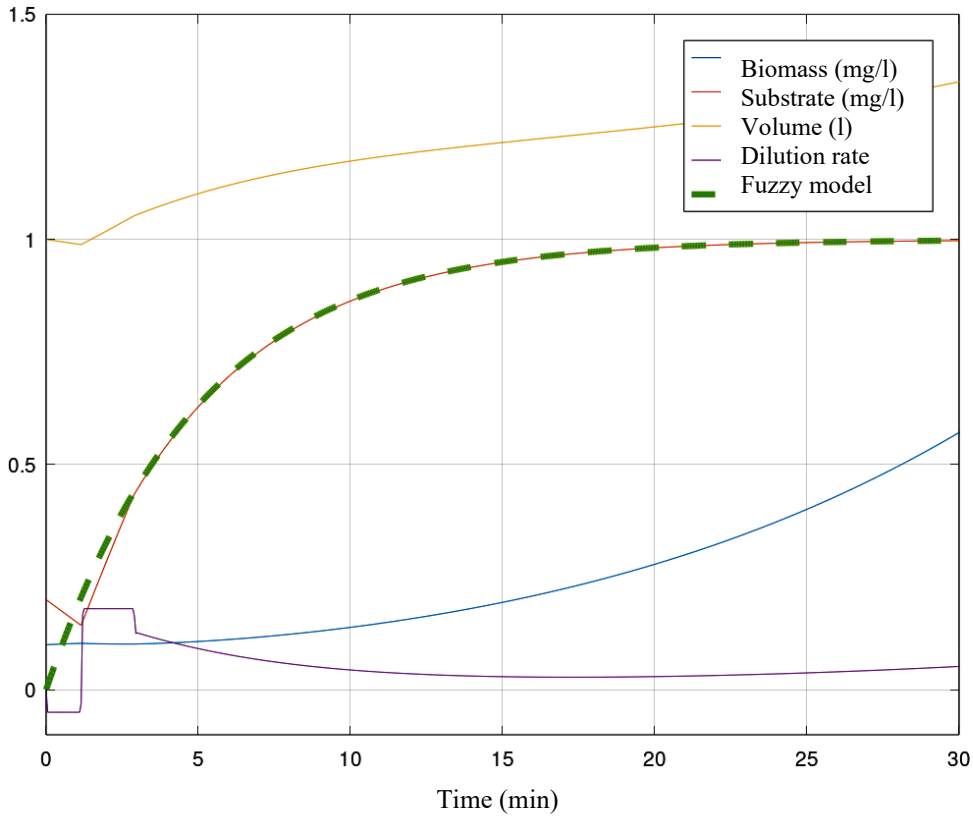

Fig. 5. Control by inverse fuzzy model of the substrate concentration.

Now, taking the vector x as input data, the initial evaluation of the network will be $A^{(0)} = x$, and for another layer ℓ we have $A^{(\ell)} = \Phi^{(\ell)}(A^{(\ell-1)})$. This function Φ is calculated with the splines $\mathbf{B}$ using:

$$\Phi^{(\ell)}(A) = \sum_{i=1}^{L} \mathbf{B}_i^{(\ell)}(A_{:,i})\, w_i^{(\ell)}, \tag{10}$$

where L is the number of layers, $\mathbf{B}_i^{(\ell)}(x)$ is the B-spline basis matrix for feature i in layer ℓ, and $w_i^{(\ell)}$ are trainable coefficients. In this way, the bases $\mathbf{B}$ used to find the polynomials that will evaluate each input are obtained, for the case of biomass $X(t)$, the bases are shown in Figure 6.

Now, the loss function is defined as:

$$\boldsymbol{L} = \frac{1}{N}\sum_{i=1}^{n}\left(Y_i - A_i^{(L)}\right)^2,$$

being Y the desired output to approach.

The coefficients $w_i^{(\ell)}$ can be initialized by least squares, thus $w_i^{(\ell)} = (\mathbf{B}_i^{(\ell)T}\mathbf{B}_i^{(\ell)} + \lambda \mathrm{I})^{-1}\mathbf{B}^{(\ell)\mathrm{T}}Y$, where λ is the Tikhonov term and serves as a balance between the error and the norm of $w^{(\ell)}$ to avoid overfitting (Samar and Lin, 2022). Then, with a finer adjustment, the weights $w_i^{(\ell)}$ can be adjusted using gradient descent as follows:

$$\nabla_{w_i^{(\ell)}} \boldsymbol{L} = -\frac{2}{n}\left(\mathbf{B}_i^{(\ell)}\right)^T\left(Y - A^{(L)}\right), \tag{11}$$

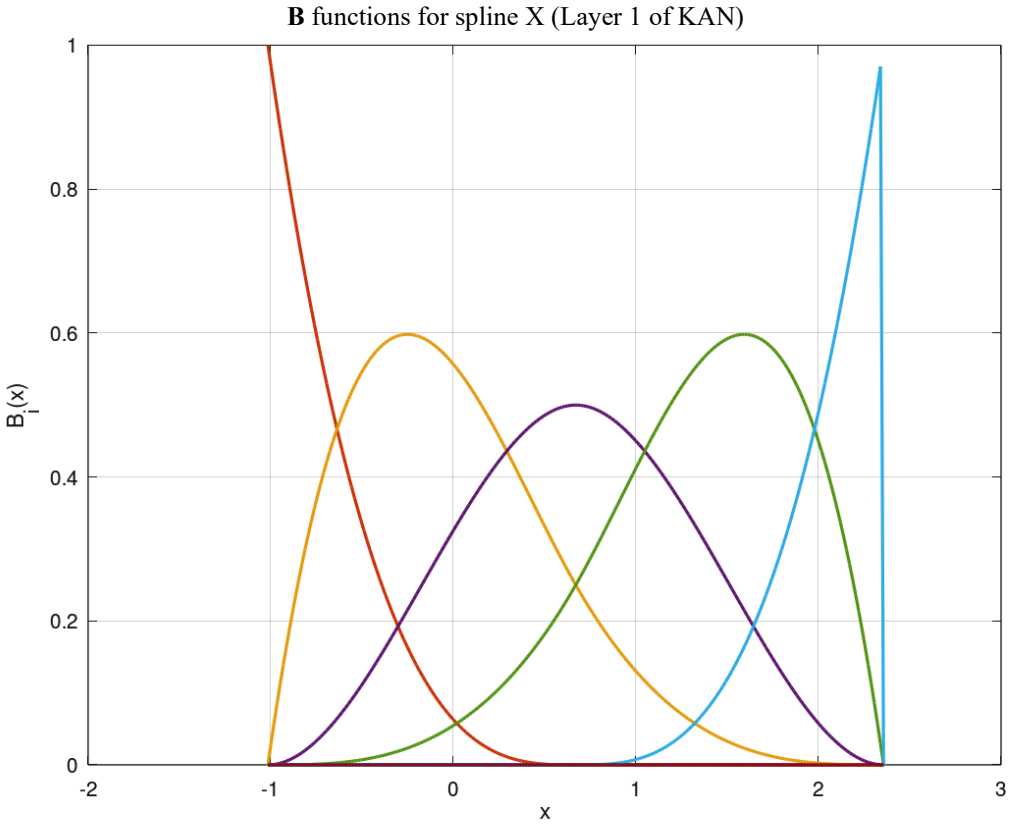

Fig. 6. Bases **B** used to granulate the input signal $X(t)$.

and thus update them with $w_i^{(\ell)} \leftarrow w_i^{(\ell)}\text{-}\eta\nabla_{wi(\ell)}\boldsymbol{L}$, where η is the training gain.

It is then possible to use some optimization method for network training that is common to ANN, such as adaptive moment estimation (ADAM) (Salihu et al., 2025). Once the bases **B** for each input are found, and the weighted sum by the weights $w_i^{(\ell)}$ is done, the polynomials that describe each variable before being aggregated in a last layer are obtained. It was proposed to use five bases per variable to still have the interpretability of the model. Similarly, better results could be obtained in the case of fuzzy modeling, but the number of rules would increase so that the interpretability and the reason for using a fuzzy model lose sense. The polynomials that describe the variables are shown in Fig. 8.

The splines found for the model variables used $D(t)$, $X(t)$ and $V(t)$ are defined by:

$$D(t) = 1.257B_1(D) - 0.059B_2(D) + 0.730B_3(D) - 1.216B_4(D) + 0.394B_5(D),$$
$$X(t) = -0.353B_1(X) + 3.168B_2(X) - 4.315B_3(X) + 2.218B_4(X) - 2.562B_5(X),$$
$$V(t) = -2.748B_1(V) + 1.595B_2(V) - 0.263B_3(V) + 0.066B_4(V) - 1.146B_5(V).$$

For the control of the bioprocess, a KAN model is trained, where the output of a layer ℓ is given by (10). Thus the approximation of the substrate concentration is the composition of the used splines given input signal:

$$S_{+1} = f(X,S,V,D) = f_X(X) + f_S(s) + f_V(V) + f_D(D),$$
$$= \sum_{i=1}^{n} w_{X,i}\mathbf{B}_i^X(X) + \sum_{i=1}^{n} w_{S,i}\mathbf{B}_i^S(X) + \sum_{i=1}^{n} w_{V,i}\mathbf{B}_i^V(X) + \sum_{i=1}^{n} w_{D,i}\mathbf{B}_i^D(X),$$

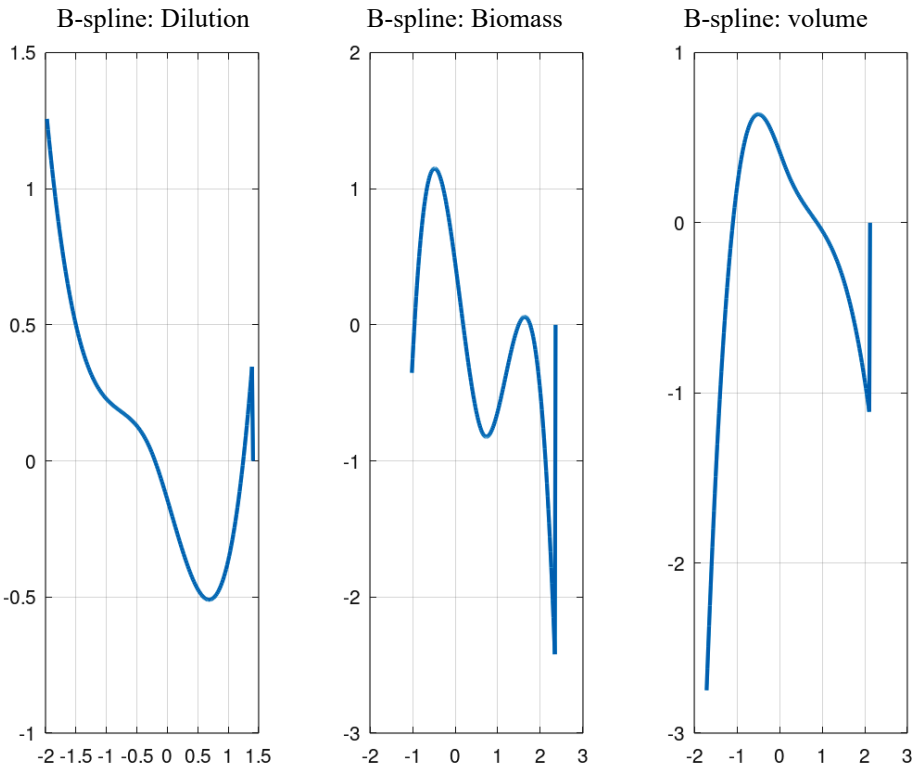

Fig. 7. Learned B-spline basis functions and their weighted contributions for each input variable in the KAN model.

where the univariate functions f are B-splines and each one shows the contribution of each variable to the found model with their respective bases **B** and weights *w*.

The model can be simplified by doing pruning in the KAN model by eliminating the axes (synaptic connections) where the weights $w_i^{(\ell)}$ are small enough to be able to omit their contribution. Once pruning is done, it would be necessary to readjust the weights of the model through gradient descent. The KAN model obtained from the bioprocess using the same dilution rate used to find the fuzzy model is shown in Figure 7.

For the inversion, the dilution rate D that would produce the concentration S_{ref} is sought. For which D^* must be found thanks to being able to use the approximate KAN model. Specifically, it was employed the Levenberg-Marquardt algorithm to solve the minimization problem given in (12), due to its efficiency in handling nonlinear least-squares problems. The numerical implementation proceeds as follows:

Given the KAN model approximation $S_{+1} = f(X, S, V, D)$, at each control interval, it is solved

$$D^* = \arg\min_D \left\| f(X, S, V, D) - S_{ref} \right\|^2, \tag{12}$$

The Jacobian matrix $J = {\partial f}/{\partial D}$ is computed analytically from the B-spline representation of the KAN model. For a B-spline of order k defined on knots $t_0, t_1, \ldots t_m$, the derivative with respect to the input variable can be efficiently computed using the recursive relation:

$$\frac{d\mathbf{B}_{i,k}(x)}{dx} = \frac{k}{t_{i+k} - t_i}\mathbf{B}_{i,k-1}(x) - \frac{k}{t_{i+k+1} - t_{i+1}}\mathbf{B}_{i+1,k-1}(x).$$

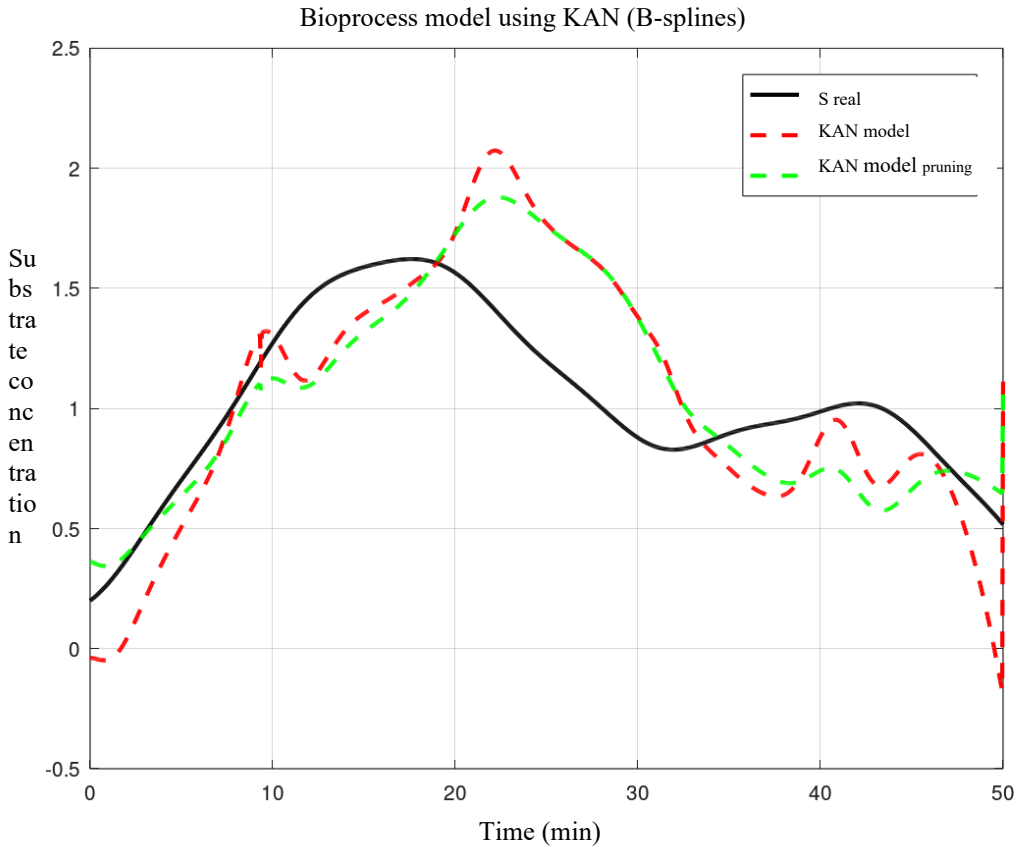

Fig. 8. Comparison of KAN model before and after pruning. The complete KAN model (solid line) utilizes all initialized nodes and connections, the pruned model (dashed line) retains only the most significant based on eliminating connections with $|w_i^{(\ell)}|<0.1$

This analytical differentiation allows for precise gradient information during optimization. The optimization routine was implemented with the following parameters:

- Initial guess: $D_0=D_{\text{previous}}$ (the dilution rate from the previous control interval)
- Convergence tolerance: 10^{-6} on the objective function
- Maximum iterations: 50 per control step
- Damping parameter λ: adaptively adjusted based on the reduction in residual

The computational burden of this online optimization is mitigated by the relatively low-dimensional input space and the locality of B-spline evaluations. Each B-spline basis function has compact support, meaning that for any given input value, only k+1 basis functions are non-zero, significantly reducing the computational cost of function evaluations and Jacobian computations.

In practice, the inversion algorithm converges within 3-8 iterations for most control intervals, with computation times suitable for real-time implementation in the studied bioprocess. The explicit functional representation provided by the KAN enables this efficient inversion, a feature not readily available in black-box neural network models.

The control is performed by inverting the KAN model, using the same reference as in the case of the inversion of the fuzzy model, the simulation shown in Fig. 9 is obtained. In this case, the D that minimizes the difference between what the KAN model gives and the proposed reference is sought.

A limitation for this technique is to use the adequate number of knots to approach polynomials in order to have an accurate KAN model.

Despite their interpretability advantages, KANs based on B-splines entail higher computational costs during training compared to traditional neural networks. This computational overhead arises from the evaluation of B-spline basis functions and the optimization of their coefficients across multiple layers. The training process requires calculating B-spline expansions for each input variable and solving linear systems for weight initialization, followed by gradient-based optimization using methods like ADAM. Furthermore, the inversion of the KAN model for control purposes, which involves solving the optimization problem in (12), adds additional computational burden as it requires iterative numerical methods to find the optimal dilution rate. However, these costs can be mitigated through strategic pruning of insignificant connections and the use of efficient B-spline evaluation algorithms. The trade-off between computational complexity and model interpretability remains a consideration for real-time implementations, though the functional transparency gained through KANs provides valuable insights for system analysis and controller design.

4 Interpretability of the Model and Control Law

One of the key benefits of using models based on Kolmogorov-Arnold networks and fuzzy systems lies in their interpretability, which makes them ideal candidates for applications where understanding the system's behavior is as important as the accuracy of its response. Additionally, the models are approximate and can be obtained by expert knowledge, or as in this case, the model parameters are found using least squares for the fuzzy case and by gradient descent for the KAN from the obtained measurements.

In the case of the fuzzy model, interpretability is manifested through its fuzzy rules that involve linguistic variables and membership functions, which directly reflect the expert knowledge or empirical logic of the system. Each rule can be understood as a statement of the type: "If the biomass is high and the volume is low, then the dilution should decrease", which has physical meaning and allows a clear qualitative analysis of the control system.

For their part, KAN benefit from the use of B-spline type functions as univariate basis functions. Unlike traditional neural networks, KAN allow directly visualizing how each input (such as biomass, dilution, or volume) contributes to the model's output. The functions obtained after training and pruning can be expressed in analytical form, which reinforces their "gray box" character. This characteristic not only facilitates model validation, but also enables its implementation in embedded control systems where transparency is essential.

Regarding the control law (dilution rate), both the fuzzy model and the KAN can be inverted to generate the control signal that produces a desired behavior. In the fuzzy model, this inversion is performed through a decomposition of the consequents and dynamic evaluation of the active rules. In the case of the KAN, an online minimization procedure is proposed where, given a reference of the desired substrate, the dilution rate D that minimizes the difference between the model's expected output and the reference is sought.

This allows adaptive and flexible control without requiring explicit models of the dynamic system, using only the knowledge encapsulated in the obtained splines.

The combination of interpretability and inversion capability makes both the fuzzy model and the KAN powerful tools for designing robust control laws, especially in nonlinear systems such as bioprocesses. Additionally, their explicit representation allows integration with other symbolic analysis techniques and formal validation.

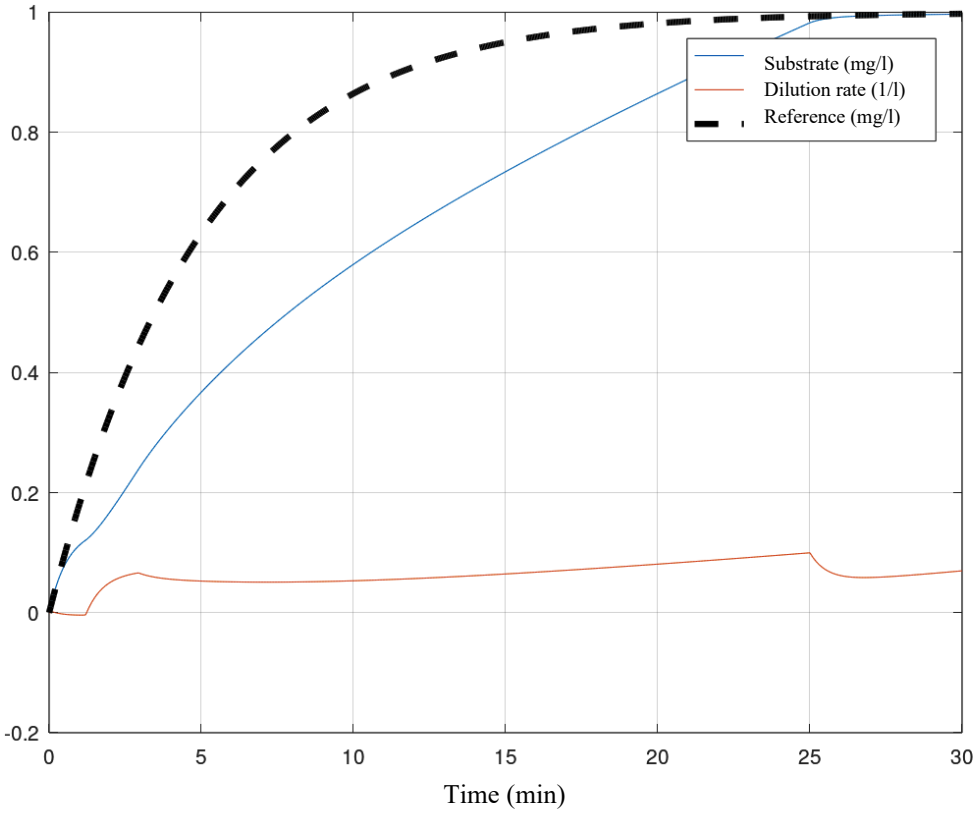

Fig. 9. Control Inverting the KAN model.

Finally, the controller performance was assessed using the Integral Time Absolute Error (ITAE) and the Integral Absolute Error (IAE) criteria, the first penalizes errors that persist over time by weighting the absolute error with time. And the second only takes into consideration the error. These criteria are defined as follows:

$$IAE = \int_0^{t_{end}} |e(t)|dt,$$

$$ITAE = \int_0^{t_{end}} t|e(t)|dt.$$

The resulting criteria evaluation using both control techniques (fuzzy logic, KAN network) are shown in Table 2.

Table 2. *Error Criteria*

	IAE	ITAE
Fuzzy model control	0.9243	0.7241
KAN model control	4.8506	3.2973

5 Conclusions

In this work, two approaches for the inverse modeling of a biotechnological system were explored: a fuzzy model based on linguistic rules and a Kolmogorov-Arnold type model trained with B-spline functions. Both models showed adequate performance in approximating the dynamic behavior of the system and allowing its inversion for control purposes. One of the main advantages of using KAN is their interpretable nature, which allows representing each input of the system through explicit univariate functions. This characteristic, absent in traditional neural networks, provides the possibility of analyzing the network's behavior, performing pruning of irrelevant nodes, and generating symbolic expressions of the model. Additionally, their compatibility with optimization methods allows implementing inverse model control laws without needing to derive complex differential expressions.

In comparison, the fuzzy model offers high interpretability thanks to its structure based on rules and membership functions, facilitating its analysis from a logical and heuristic perspective. Both approaches are viable for control applications in nonlinear biological systems. However, the approach with KAN presents an additional promise: it combines the functional expressiveness of neural networks with the transparency of symbolic systems, which is crucial in environments where not only precision is required, but also explainability and model traceability. In future works, the extension of the model to multiple layers will be explored, as well as the implementation of KAN control in embedded hardware and its comparison with modern techniques such as LSTM networks or fuzzy-KAN hybrid models. Additionally, research will focus on enhancing robustness to measurement noise, conducting comprehensive sensitivity analyses, and optimizing computational efficiency for real-time applications.

6 References

Ammara, U., Zehra, S. S., Nazir, S., & Ahmad, I. (2024). Artificial neural network-based nonlinear control and modeling of a DC microgrid incorporating regenerative FC/HPEV and energy storage system. *Renewable Energy Focus, 49*, 100565. https://doi.org/10.1016/j.ref.2024.100565

Aslam, M. S., & Bilal, H. (2024). Modeling a takagi-sugeno (T-S) fuzzy for unmanned aircraft vehicle using fuzzy controller. *Ain Shams Engineering Journal, 15*(10), 102984. https://doi.org/10.1016/j.asej.2024.102984

Bárcenas-Castañeda, M., Calatayud-Velázquez, L. E., Roblero-Aguilar, S. S., Solís-Romero, J., & Castellanos-Escamilla, V. A. (2023). Expert system through a fuzzy logic approach for the macroscopic visual analysis of corroded metallic ferrous surfaces: Knowledge acquisition process. *Expert Systems with Applications, 214*, 119104. https://doi.org/10.1016/j.eswa.2022.119104

Hong, Y., Fu, C., & Merci, B. (2023). Optimization and determination of the parameters for a PID based ventilation system for smoke control in tunnel fires: Comparative study between a genetic algorithm and an analytical trial-and-error method. *Tunnelling and Underground Space Technology*, *136*, 105088. https://doi.org/10.1016/j.tust.2023.105088

Kassem, Y., & Camur, H. (2017). Prediction of biodiesel density for extended ranges of temperature and pressure using adaptive neuro-fuzzy inference system (ANFIS) and radial basis function (RBF). *Procedia Computer Science, 120*, 311–316. https://doi.org/10.1016/j.procs.2017.11.244

Khosravi, H., & Bahram, M. (2025). Prediction of geopolymer concrete compressive strength using artificial neural network and genetic algorithm. *Results in Engineering*, *27*, 105537. https://doi.org/10.1016/j.rineng.2025.105537

Konar, A. (2000). *Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain*. CRC Press.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., & Tegmark, M. (2024). *KAN: Kolmogorov–Arnold networks* (arXiv:2404.19756). arXiv. https://arxiv.org/abs/2404.19756

Márquez-Vera, C. A., Márquez-Vera, M. A., Yakoub, Z., Ma'arif, A., Castro-Montoya, A. J., & Cázarez-Castro, N. R. (2022). Fuzzy state feedback with double integrator and anti-windup for the Van de Vusse reaction. *Archives of Control Science, 32*(2), 383–408.

Márquez-Vera, M. A., Ramos-Fernández, J. C., Cerecero-Natale, L. F., Lafont, F., Balmat, J.-F., & Esparza-Villanueva, J. I. (2016). Temperature control in a MISO greenhouse by inverting its fuzzy model. *Computers and Electronics in Agriculture, 124*, 168–174. https://doi.org/10.1016/j.compag.2016.04.005

Nguyen, A. T., Nguyen, N. H., & Trinh, M. L. (2025). Fuzzy PD control for a quadrotor with experimental results. *Results in Control and Optimization, 19*, 100568.

Ouifak, H., & Idri, A. (2025). A comprehensive review of fuzzy logic based interpretability and explainability of machine learning techniques across domains. *Neurocomputing*, *647*, 130602. https://doi.org/10.1016/j.neucom.2025.130602

Salihu, S. A., Adebayo, S. O., Abikoye, O. C., Usman-Hamza, F. E., Mabayoje, M. A., Brahma, B., & Bandyopadhyay, A. (2025). Detection and classification of potato leaves diseases using convolutional neural network and ADAM optimizer. *Procedia Computer Science, 258*, 2–17. https://doi.org/10.1016/j.procs.2025.04.159

Samar, M., & Lin, F. R. (2022). Perturbation analysis and condition numbers for the Tikhonov regularization of total least squares problem and their statistical estimation. *Journal of Computational and Applied Mathematics, 411*, 114230. https://doi.org/10.1016/j.cam.2022.114230

Searle, J. R. (1990). Cognitive Science and the Computer Metaphor. *Artificial Intelligence, Culture and Language: On Education and Work*, 23-34. The Springer Series on Artificial Intelligence and Society, London. https://doi.org/10.1007/978-1-4471-1729-2_4

Surma, R., Wojcieszyńska, D., Mulla, S. I., & Guzik, U. (2025). Current strategy of non-model-based bioprocess optimizations with genetic algorithms in bioscience - a systematic review. *Computers in Biology and Medicine, 192-A*, 110247. https://doi.org/10.1016/j.compbiomed.2025.110247

Tanaka, K., & Wang, H. O. (2001). *Fuzzy control systems design and analysis*. John Wiley & Sons, Inc., New York.

Thakolkaran, P., Guo, Y., Saini, S., Peirlinck, M., Alheit, B., & Kumar, S. (2025). Can KAN CANS? input-convex Kolmogorov-Arnold networks (KANS) as hyperelastic constitutive artificial neural networks (CANS). *Computer Methods in Applied Mechanics and Engineering, 443*, 118089. https://doi.org/10.1016/j.cma.2025.118089

Wang, F. (2015). Simulation tool for asynchronous cortical streams (STACS): Interfacing with spiking neural networks. *Procedia Computer Science, 61*, 322–327. https://doi.org/10.1016/j.procs.2015.09.149

Wang, G., Li, X., & Xie, S. (2024). Bilateral Turing test: Assessing machine consciousness simulations. *Cognitive Systems Research, 88*, 101299. https://doi.org/10.1016/j.cogsys.2024.101299