Adaptive PID Tuning Using Kolmogorov-Arnold Networks

Lily Chiparova *, Vasil Popov, Sevil Ahmed-Shieva *, Nikola Shakev

Posted Date: 30 April 2026

doi: 10.20944/preprints202604.2157.v1

Keywords: Kolmogorov-Arnold networks; PID; adaptive control; neural networks

 doi:10.20944/preprints202604.2157.v1

Article

Adaptive PID Tuning Using Kolmogorov-Arnold Networks

Lily Chiparova *, Vasil Popov, Sevil Ahmed-Shieva * and Nikola Shakev

Control Systems Department, Faculty of Electronics and Automation, Technical University Sofia, Branch Plovdiv, 4000 Plovdiv, Bulgaria

* Correspondence: lili.shiparova@abv.bg (L.C.); sevil.ahmed@tu-plovdiv.bg (S.A.)

Abstract

The paper proposes an implementation of Kolmogorov-Arnold networks (KANs) for the purpose of dynamic proportional-integral-derivative (PID) control tuning in first- and second-order linear systems under noisy and time-varying reference conditions. Toy datasets, based on instantaneous system error, output and reference trajectory, are used for training the networks and comparing KANs results over a performance of: i) a PID with fixed coefficients, taken from MATLAB's Simulink PID Autotune; ii) an MLP based neural network (NN), trained on the same datasets; iii) a traditional adaptive PID scheme with gain scheduling; iv) an LMS-based online tuning approach. Results show that KANs outperform MLPs and LMS even with less optimized datasets under noisy and quick-changing conditions and perform on par with methods, such as gain scheduling, while allowing for more flexibility and easier setup.

Keywords: Kolmogorov-Arnold networks; PID; adaptive control; neural networks

1. Introduction

PID controllers are a benchmark, when it comes to control in both industrial and scientific settings. First introduced over a hundred years ago, their simplicity, robustness and ease of implementation alongside their relative inexpensiveness result in near 90% use when it comes to closed-loop control tasks (industrial control tasks) and taught in all universities and technical colleges [1,2]. Some examples of PID use in different industries include, but are not limited to: the chemical industry, where PID controllers are widely used for temperature control, pressure and flow regulation, concentration control and pH level stabilisation, in energy and power generation industries for stabilizing power output, in robotics and motion control for industrial robots joint position control, in temperature control cases for managing electrical furnaces thermal response, managing reactor temperatures in nuclear power systems, thermal management of electrical vehicle batteries and more [3–7].

But their successful implementation relies on proper tuning of its parameters K_p, K_i and K_d. While classical tuning rules, such as Ziegler-Nichols, Cohen- Coon and Lambda tuning are exceptional when working with fixed, idealized plants and only narrow class of systems, they often fail in practical settings where system parameters change, disturbances occur, sensor introduce noise into measurements and loads vary over time [8].

Adaptive PID control has been extensively explored as a solution for the inherent limitations of fixed-gain PID controllers in the presence of disturbances, noise, and time-varying operating conditions. Existing adaptive schemes can generally be grouped into model-based methods, evolutionary optimization techniques, and learning-based approaches. Model-based approaches rely on accurate parametric models of the plant—such as linearized or reduced-order models—to compute gain updates in real time [9,10]. However, these methods are often difficult to deploy in practical applications due to unmodeled dynamics, parameter drift, and the challenge of obtaining high-fidelity system models [11]. Evolutionary optimization techniques, including genetic algorithms

 doi:10.20944/preprints202604.2157.v1

and particle-swarm optimization, provide global search capabilities but generally lack the computational speed needed for real-time online tuning [12,13]. Learning-based adaptive PID controllers leverage neural networks or data-driven mappings to update gains without explicit models [14], but their success depends strongly on high-quality training data and tuned architectures. These limitations have motivated the development of more advanced machine-learning techniques capable of supporting automatic, data-efficient, and online-capable PID tuning for complex dynamical systems.

In recent years some great works exploring alternative PID tuning methods have been proposed. An MLP-based approach for tuning a PID controller in high-temperature system was developed and tested in real-life working conditions in [15]. Another study focuses on implementing reinforcement learning for automatic PID tuning while maintaining closed-loop stability [16]. A special PIDNN was proposed for parameter tuning in [17].

Since the publication of the work outlining the edge Kolmogorov-Arnold networks have over traditional MLPs when it comes to parameter efficiency and interpretability capability, the popularity of KANs has risen in the previous year and a half. Many studies exploring KAN implementations for different purposes have been published. The architecture has been used in works exploring efficient time-series forecasting, interpretable state estimations in power systems, modeling electricity demands and physics-informed frameworks for solving forward and inverse problems with the use of KANs to name a few [18–21].

Due to the architecture's inherent interpretability and ability to approximate outputs from sparse data we believe they might offer new advantages in the field of adaptive tuning. We propose a Kolmogorov-Arnold Network- based PID tuning method as a new alternative within the class of neural-network-based methods. This class has seen extensive exploration in recent years with numerous different studies implementing conventional MLP-based networks, special PID neural networks and other related learning-based controllers [15,17]. It is trained on synthetic datasets that include multiple reference trajectories, noise levels, and plant parameter variations. The models learn a direct mapping from system state and reference features to optimal PID gains, representing modern learning-based adaptive control, and are then tested in an online simulation with time-varying system parameters, noise levels and reference trajectories.

In this paper, we design and evaluate a KAN- based PID tuning against representative techniques from the major classes of methods, mentioned above. First, we implement a classical gain-scheduling controller, representing structured model-based design where PID gains are switched or interpolated based on operating regions (e.g., reference amplitude or frequency). Second, we include a fixed-gain PID controller with parameters obtained from MATLAB's Simulink Auto-Tuner, serving as a baseline for non-adaptive control. Third, we implement an LMS-based online tuner, which belongs to gradient-based learning methods and updates gains through instantaneous error-minimization. Finally, we compare with a machine-learning-based adaptive tuner based on the multilayer perceptron (MLP) trained on the same conditions as the KAN models. By comparing these methods across both first-order and second-order linear systems, we establish a systematic performance benchmark spanning the full spectrum of classical, evolutionary-inspired, and machine-learning-based adaptive PID design by tracking performance, robustness to noise, convergence behaviour and parameter-efficiency.

2. Materials and Methods

2.1. Kolmogorov–Arnold Network (KAN) PID Tuning

2.1.1. Theoretical Base and Structure of Networks

The base of Kolmogorov-Arnold Networks is the Kolmogorov-Arnold representation theorem (KAT) stating that any smooth and continuous multivariate function can be represented as a combination of univariate continuous functions and addition [22]:

 doi:10.20944/preprints202604.2157.v1

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

where $\phi_{q,p} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$. Each $\phi_{q,p}$ is a univariate function, and the outer functions Φ_q combine the inner univariate components to reconstruct $f(x)$.

The structure of KAN is an extension of the KAT to arbitrary depth and width, made by stacking individual layers. Eq (2) gives us the output of a KAN network with L layers and input vector $\boldsymbol{x}_0 \in \mathbb{R}^{n_0}$.

$$KAN(x) = (\boldsymbol{\Phi}_{L-1} \circ \boldsymbol{\Phi}_{L-2} \circ \dots \circ \boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_0)x \quad (2)$$

Each layer with n_{in}-dimensional inputs and n_{out}-dimensional outputs can be described as a matrix of 1D functions

$$\boldsymbol{\Phi} = \{\phi_{q,p}\}, \qquad p = 1,2, \dots, n_{in}, \qquad q = 1,2, \dots, n\_out \quad (3)$$

where the activation functions $\phi(x)$ are parametrized as a linear combination of B-splines and a basis function [23]:

$$\phi(x) = w_b b(x) + w_s spline(x) \quad (4)$$

$$b(x) = silu(x) = \frac{x}{1 + e^{-x}} \quad (5)$$

$$spline(x) = \sum_i c_i B_i(x) \quad (6)$$

The structure and mathematical base of KANs is what distinguishes them from widely used MLP networks and provides an advantage when it comes to interpretability, extracting meaningful results from sparse data and number of parameters needed. KANs manage to approximate multivariate mappings with far fewer parameters and smoother representations, making them more sample-efficient for control tasks.

2.1.2. KANs for PID Gain Prediction

We explore KAN's potential for the task of adaptive PID tuning. Our networks learn the mapping $f_{KAN}: (e_t, \dot{e}_t, \int e_t) \to (K_P, K_I, K_D)$ so that the PID control law drives the system output to the reference.

$$f_{KAN}: \left(e_t, \dot{e}_t, \int e_t\right) \to (K_P, K_I, K_D) \quad (7)$$

$$u_t = K_P e_t + K_I \int e_t + K_D \dot{e}_t \quad (8)$$

KANs represent nonlinearities explicitly on the edges through the spline-based functions. Each connection between the nodes corresponds to a univariate spline that transforms one input variable before performing addition on nodes. In our PID adaptive tuning case these spline functions represent how variations in error dynamics influence the proportional, integral and derivative gains. The final layer of the networks gives us three independent outputs, which directly correspond to each PID coefficient - K_P, K_I, K_D.

Tests were performed for first- and second-order linear systems with changing levels of white Gaussian noise on both the input and output of the plants and changing reference signal magnitude and type. On Figure 1 can be seen a block diagrams for our system configuration.

 doi:10.20944/preprints202604.2157.v1

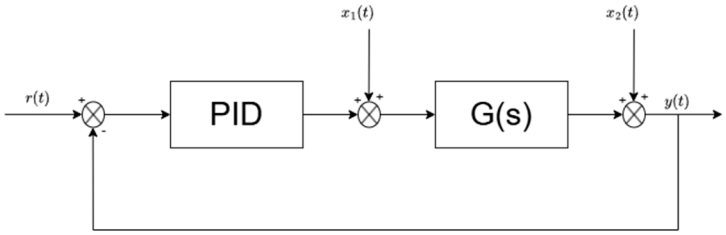


Figure 1. Block diagram of tested system. r(t) -reference signal; $x_1(t)$ - disturbance on plant input; $x_2(t)$ - disturbance on plant output; y(t) -system output; G(s)- plant transfer function. G(s) is the transfer function of either a first- or second-order linear system- $G(s) = \frac{K}{\tau s+1}$ or $G(s) = \frac{K\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$.

2.1.3. Datasets

Our datasets are synthetically generated and catered to each specific case. A summary of each dataset case can be seen in Table 1. To train both the MLP- and KAN-based PID tuning networks, we generated a unified family of synthetic datasets covering multiple operating scenarios for first- and second-order dynamic systems. All datasets share a common structure in which each sample consists of a short simulation rollout paired with the corresponding optimal PID gains (K_p, K_i, K_d). The input vectors always include the instantaneous tracking error $e[i]$, the measured plant output $y_{\text{meas}}[i]$, a normalized time index t/T_{final}, and the initial input bias U_0. Additional features are included depending on the dataset variant: for plant-varying datasets, the true plant parameters (K, τ) for first-order models or (ω_n, ζ) for second-order models are appended; in the reference-varying datasets, the instantaneous reference value $r[i]$ and its derivative $\dot{r}[i]$ are also incorporated. Across all variants, the PID gains used during dataset creation are sampled uniformly from predefined ranges appropriate for each system order. Noise conditions are systematically varied using several actuator noise levels σ_{act} and measurement noise levels σ_{meas}, enabling the networks to learn robust gain mappings. For both first-order and second-order plants, three dataset families are generated: (1) datasets with only noise variation over time, (2) datasets combining noise variation with randomized plant parameters, and (3) datasets with time-varying reference signals such as steps, sinusoids, and ramps. Each dataset contains 200–300 simulated trajectories with diverse combinations of plant parameters, reference dynamics, and noise realizations. This unified dataset design ensures that the learning models encounter a broad distribution of operating conditions, enabling them to generalize PID tuning behaviour across plant uncertainties, reference regimes, and measurement/actuation disturbances.

Table 1. Characteristics of different kinds of datasets used for training MLP and KAN models.

System	Dataset Type	Input Features	Dataset Parameters
First-order	Noise-changing only	[err, y_{meas}, $\frac{t}{T_{final}}$, U_0]	PID ranges: $K_p(0.5-35)$, K_i $(0.1-7)$, K_d $(0.01-5)$; U_0 = [2.0,18.0,6.7]; Noise σ_{act} = [0.01,0.3,0.6], σ_{meas} = [0.1,0.03,0.4]; n = 200; Variants: no noise / input-only / output-only / both
First-order	Noise + plant-changing	[err, y_{meas}, $\frac{t}{T_{final}}$, U_0, (K, τ)]	Same PID ranges; U_0 = [2,18,6.7];

 doi:10.20944/preprints202604.2157.v1

			Noise as above; τ $(7-20), (1-7)$; $n = 200$ PID: $K_P(0.5-45)$, K_i $(0.1-30)$, K_d $(0-15)$;
First-order	Reference = function	[err, y_{meas}, $\frac{t}{T_{final}}$, r,$\dot{r}$]	References: sin/ramp/step; Noise: $\sigma = [0,0.02,0.05]$; $n = 300$; $\tau = 10, K = 1$; PID K_p $(0.5-50)$, K_i $(0.5-30)$, K_d $(0.1-25)$;
Second-order	Noise-changing only	[err, y_{meas}, $\frac{t}{T_{final}}$, U_0,]	$U_0 = [2,18,6.7]$; Noise: $\sigma_{act} = [0.01,0.3,0.6]$, $\sigma_{meas} = [0.1,0.03,0.4]$; ω_n range $0.5-2.0$, ζ $0.2-1.0$; $n = 200$ PID: K_p $(0.5-50)$,, K_i $(0.5-30)$, K_d $(0.1-25)$;
Second-order	Noise + plant-changing	[err, y_{meas}, $\frac{t}{T_{final}}$, U_0, (ω_n, ζ)]	$U_0 = [2,18,6.7]$; Noise: $\sigma_{act} = [0.01,0.3,0.6]$, $\sigma_{meas} = [0.1,0.03,0.4]$; ω_n $(0.5-2.0)$, ζ $(0.2-1.0)$; $n = 200$ PID: K_p $(0.5-30)$, K_i $(0-15)$, K_d $(0-20)$;
Second-order	Reference = function	[err, y_{meas}, $\frac{t}{T_{final}}$, r,$\dot{r}$ ω_n, ζ]	ω_n $(0.5-2.0)$, $\zeta(0.2-0.8)$; References: sin/ramp/step; Noise $\sigma = [0,0.01,0.05]$; $n = 300$; $T_{final} = 25$

Figure 2 shows some dataset preview cases for first-order systems with changing noise and plant parameters.

 doi:10.20944/preprints202604.2157.v1

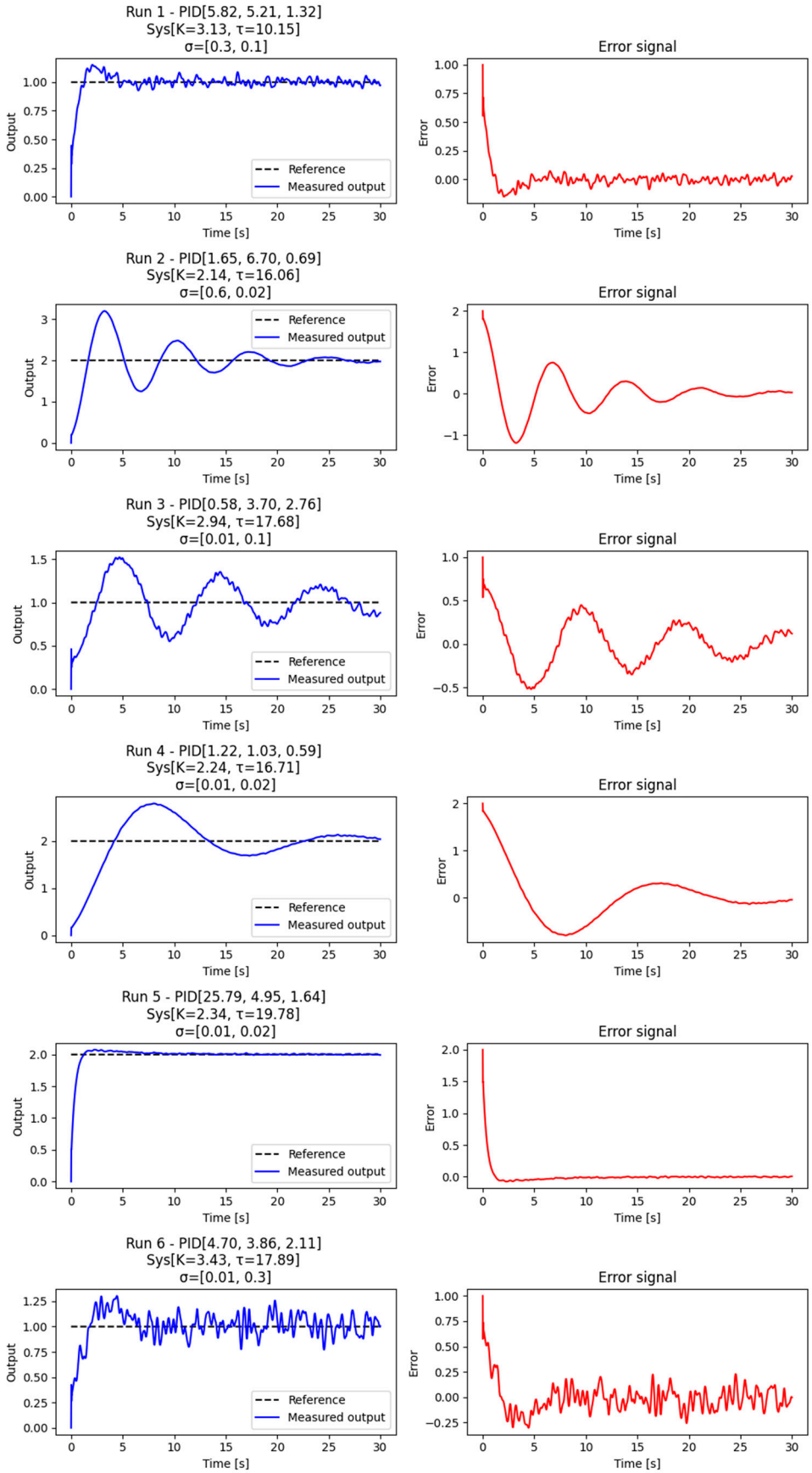


Figure 2. Dataset preview cases for first-order systems with different noise levels and plant parameters.

2.1.4. Network Structures and Training Parameters

 doi:10.20944/preprints202604.2157.v1

Across all experimental conditions, Kolmogorov–Arnold Networks were configured using compact architectures tailored to the dimensionality of each dataset variant. The open-source Pykan library was used. A summary can be seen in Table 2. For first-order systems with time-varying noise—both with and without explicit plant-parameter inputs—the KAN models used a layer structure of [4,4,4,3], corresponding to four input features and three spline-based output heads representing (K_p, K_i, K_d). These models were initialized with grid size 3 and spline order 3, and trained using progressive grid extension up to size 30 to improve local function resolution; no regularization was applied, and training used a batch size of 1000 with mean-squared error (MSE) loss.

The same architectural setup was employed for first-order datasets with both noise and plant-parameter variation. For the reference-varying first-order datasets, where the input dimensionality increases due to the inclusion of $r(t)$ and $\dot{r}(t)$, the network structure was expanded to [5,4,4,3]. These models used the same spline order but only a single extension step from grid 3 to grid 5, reflecting the reduced need for fine-grained partitioning under smooth reference signals. For second-order systems, the noise-only and noise-plus-plant-variation datasets both used the [4,4,4,3] structure without grid extension, again with spline order 3, no regularization, and MSE loss. Finally, for the second-order reference-varying datasets, the input dimensionality increases due to inclusion of $(r, \dot{r}, \omega_n, \zeta)$, so the network was expanded to [7,4,4,3] and trained with an extension from grid size 3 to 5. All KAN models were trained on full datasets, and no sparsity or L1/L2 penalties were used, ensuring that performance differences arise strictly from dataset complexity rather than architectural biases.

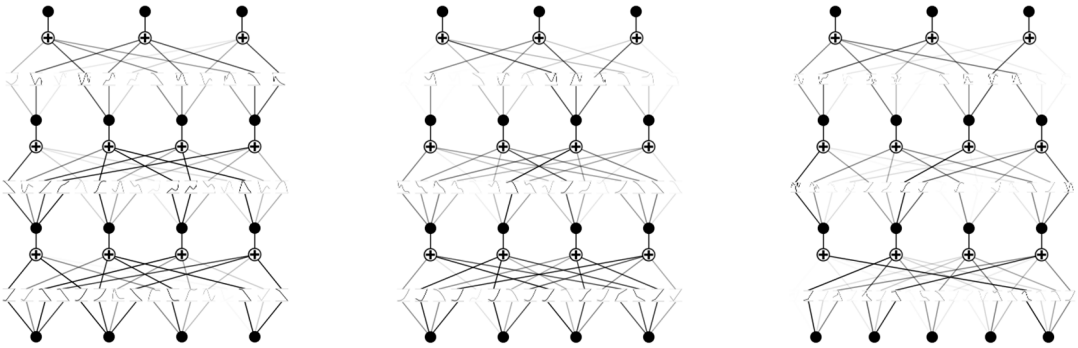
Table 2. KAN models architecture and parameters.

System Order	Dataset Variant	Input Dim.	KAN Shape	Grid Size (Start → End)	Spline Order	Regularization	Batch Size	Loss
1st-order	Noise variation only	4	[4,4,4,3]	3 → 30	3	None	1000	MSE
1st-order	Noise + plant variation	4 (+2 if included)	[4,4,4,3]	3 → 30	3	None	1000	MSE
1st-order	Reference-varying inputs	5	[5,4,4,3]	3 → 5	3	None	1000	MSE
2nd-order	Noise variation only	4	[4,4,4,3]	3 → 3	3	None	Full dataset	MSE
2nd-order	Noise + plant variation	4 (+2 if included)	[4,4,4,3]	3 → 3	3	None	Full dataset	MSE

 doi:10.20944/preprints202604.2157.v1

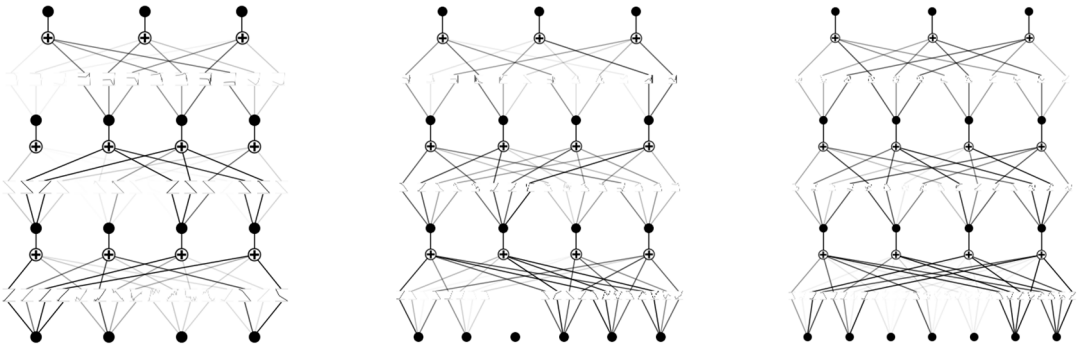
2nd-order	Reference-varying inputs	7	[7,4,4,3]	3 → 5	3	None	1000	MSE

Figures 3 and 4 provide a visualization of the trained KAN models for first- and second-order system cases. Each plot shows the layered spline structure of the network. Darker connections indicate stronger functional sensitivity. For first-order models smoother and more uniformly distributed spline activations are present, than in second-order cases, which is to be expected, as the spline connections are a reflexion of the error dynamics of each plant variation.



(**a**) (**b**) (**c**)

Figure 3. KAN model plots for first-order plants: (**a**) Model shape for varying noise level datasets; (**b**) Model shape for varying noise levels and plant parameters; (**c**) Model shape for time-varying reference signal datasets.



(**a**) (**b**) (**c**)

Figure 4. KAN model plots for second-order plants: (**a**) Model shape for varying noise level datasets; (**b**) Model shape for varying noise levels and plant parameters; (**c**) Model shape for time-varying reference signal datasets.

2.2. MLP Networks for PID Tuning

Multilayer perceptrons (MLPs) serve as a strong baseline for comparison against KANs because they represent the standard feedforward neural architecture widely used for nonlinear system modeling and controller tuning. MLPs have been previously applied to adaptive PID gain synthesis—for example, in high-temperature thermal control systems—demonstrating good approximation capability and robustness across operating conditions [15]. In this work, all MLPs

 doi:10.20944/preprints202604.2157.v1

were trained on the same datasets used for the KAN experiments, ensuring a direct and fair comparison. Furthermore, training and implementation were carried out using the Pykan framework, leveraging its unified interface for both KAN and standard neural network modules.

Across all dataset variants, MLP architectures were designed to match the dimensionality of the input features while providing sufficient depth and width to capture nonlinear mappings from system states to PID gains. As can be seen in Table 3, all models employed ReLU activation functions, the LBFGS optimizer, learning rate 1×10^{-4}, batch size 1000, and an MSE loss function.

Table 3. MLP models architecture and parameters.

System Order	Dataset Variant	Input Dim.	MLP Shape	Activation	Optimizer	LR	Batch Size	Loss
1st-order	Noise variation only	4	[4, 64, 64, 32, 3]	ReLU	LBFGS	1e-4	1000	MSE
1st-order	Noise + plant variation	4 (+2 optional)	[4, 64, 64, 32, 3]	ReLU	LBFGS	1e-4	1000	MSE
1st-order	Reference-varying	5	[5, 64, 64, 32, 3]	ReLU	LBFGS	1e-4	1000	MSE
2nd-order	Noise variation only	4	[4, 128, 128, 64, 3]	ReLU	LBFGS	1e-4	1000	MSE
2nd-order	Noise + plant variation	4 (+2 optional)	[4, 128, 128, 64, 3]	ReLU	LBFGS	1e-4	1000	MSE
2nd-order	Reference-varying	7	[7, 64, 64, 32, 3]	ReLU	LBFGS	1e-4	1000	MSE

2.3. LMS- Based Online PID Tuning

The next method we compared our KAN models to is an LMS-based online tuner. While more recent adaptive PID work often uses reinforcement learning or model-free optimization, classical LMS-based PID tuning remains relevant. Some examples include a hybrid LMS and neural network controller for PID gain adaptation, a self-learning PID with three independent LMS updates for K_p, K_I and K_D, which showcases the algorithm's low computational expense [24,25].

Our implementation adjusts the gains K_P, K_I and K_D online using an error-driven learning rule. The instantaneous tracking error is continuously minimized:

$$e(t) = r(t) - y(t). \tag{9}$$

The PID gains are updated at every sampling step, which allows for compensation of disturbances, noise, trajectory changes and variations of system dynamics.

A PID controller can be expressed as a linear combination of three signals:

$$u(t) = K_P e(t) + K_I x_I(t) + K_D x_D(t) \tag{10}$$

where $x_I(t) = \int_0^t e(\tau)\, d\tau$ is the integral of the error and $x_D(t) = \frac{de(t)}{dt}$ is the derivative of the error. The same expression in vector form becomes:

$$u(t) = \boldsymbol{w}^T \boldsymbol{x}(t) \tag{11}$$

where

$$\boldsymbol{x}(t) = \begin{bmatrix} e(t) \\ x_I(t) \\ x_D(t) \end{bmatrix}, \boldsymbol{w}(t) = \begin{bmatrix} K_P(t) \\ K_I(t) \\ K_D(t) \end{bmatrix}. \tag{12}$$

We can see that this is identical to the structure of a single-node neural network(ADALINE) and tuning a PID controller is in practice equivalent to adjusting the weights of a linear neuron [26].

For our tuning method we define an instantaneous quadratic loss function:

 doi:10.20944/preprints202604.2157.v1

$$L(t) = \frac{1}{2}e(t)^2. \tag{13}$$

where the goal is minimization of this function. That's a standard setting for LMS methods. The LMS update law originates from the Wiener filtering problem, where the goal is to find a set of PID gains $\boldsymbol{w} = [K_P, K_I, K_D]^T$ that minimize the mean-squared tracking error [27]. In the ideal case, the optimal gains satisfy the Wiener equation $\boldsymbol{Rw} = \boldsymbol{p}$, where $\boldsymbol{R} = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T]$ is the autocorrelation matrix of the PID basis signals $x(t) = [e(t), x_I(t), x_D(t)]^T$, and $\boldsymbol{p} = \mathbb{E}[r(t)x(t)]$ is their cross-correlation with the reference. Since these expectations cannot be computed in real time, LMS replaces them with instantaneous samples, leading to the stochastic gradient descent update:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \eta\, e(t)\, \boldsymbol{x}(t), \tag{14}$$

which adjusts the parameters along the negative gradient of the squared error surface. For the PID controller, this yields separate update rules for each gain:

$$\begin{aligned} K_P(t+1) &= K_P(t) + \eta\, e(t)\, e(t), \\ K_I(t+1) &= K_I(t) + \eta\, e(t)\, x_I(t), \\ K_D(t+1) &= K_D(t) + \eta\, e(t)\, x_D(t). \end{aligned} \tag{15}$$

Each gain is therefore nudged in the direction that locally reduces tracking error based solely on measured signals, requiring no model of the plant. This makes LMS-PID tuning computationally light, adaptive, and well suited for real-time control applications.

Algorithm 1 presents the LMS-based online PID adaptation scheme used in our work. The PID gains and the integral state and previous error are initialized in lines 1 and 2. At each control step, the plant output and reference signal are measured (lines 3 and 4) and the tracking error is computed as shown in line 5. The integral and derivative basis signals are then updated in lines 6 and 7 respectively. Line 8 forms the control input by using the updated basis signals, which is then applied to the plant in line 9. An update of the PID gains follows using the LMS laws in line 10. These updates adjust each gain proportionally to the instantaneous tracking error and its corresponding basis signal, implementing a gradient-based minimization of the squared error.

Algorithm 1. LMS-Based Online PID Adaptation

Step	Description
1	Initialize PID gains $K_P, K_I,$ and K_D.
2	Initialize integral state $x_I = 0$ and previous error $e_{prev} = 0$.
3	Loop at each sampling step:
4	Measure plant output $y(t)$ and read reference signal $r(t)$.
5	Compute tracking error: $e(t) = r(t) - y(t)$.
6	Update integral basis signal: $x_I = x_I + e(t)\Delta t$.
7	Update derivative basis signal: $x_D = \frac{e(t) - e_{prev}}{\Delta t}$.
8	Compute control input: $u(t) = K_P e(t) + K_I x_I + K_D x_D$
9	Apply control input u(t) to the plant
10	Update PID gains using LMS adaptation: $K_P = K_P + \eta e(t) e(t)$ $K_I = K_I + \eta e(t) x_I$ $K_D = K_D + \eta e(t) x_D$
11	Optional: Clip gains to predefined safety bounds.
12	Update previous error: $e_{prev} = e(t)$.
13	End Loop

An optional constrain of the adapted gains can be added in line 11 to ensure safe operation of the controller. Finally, in line 12 is updated the previous error and the procedure is repeated for each control cycle – line 13.

 doi:10.20944/preprints202604.2157.v1

2.4. Fixed PID Controller (Baseline)

A conventional PID controller computes the control action as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}, \quad (16)$$

where $e(t) = r(t) - y(t)$.

This baseline assumes time-invariant gains K_p, K_i, K_d that do not adapt to disturbances or plant variations. Gains were obtained using MATLAB Simulink Auto-Tuner (classical frequency-domain tuning procedure).

A first-order derivative filter was used to suppress measurement noise.

$$D(s) = K_d \frac{N}{1 + N/s}, N = 80, \quad (17)$$

No model training is involved. The fixed PID is directly evaluated under:

- step, ramp, and sinusoidal reference trajectories;
- varying actuator and measurement noise;
- plant parameter variations (first- and second-order systems)

2.5. Gain Scheduling PID Controller

Gain scheduling uses different PID gains depending on a measured scheduling variable σ(t). Classical theory defines:

$$K_p(t) = f_p(\sigma(t)), K_i(t) = f_i(\sigma(t)), K_d(t) = f_d(\sigma(t)), \quad (18)$$

where f_p, f_i, f_d are lookup tables or interpolation maps over operating points, determined by the scheduling variable [28,29].

In this work the scheduling variable is the reference amplitude or reference frequency, depending on the input type.

The gain table contains three operating regions: small inputs, moderate inputs, high inputs. Gains are interpolated smoothly using:

$$K(\sigma) = \alpha(\sigma)K_1 + \big(1 - \alpha(\sigma)\big)K_2. \quad (19)$$

The same simulation environment is used as in Section 2.4. No training is required; gains are manually selected based on plant response.

2.6. Evaluation Metrics

The results obtained in this work were evaluated both graphically and numerically. A three-segment simulation test consisting of three consecutive 10-second time intervals with different reference signals, noise levels, and plant parameters was used for performance trials of each control strategy.

We provide plot comparisons of the performance of each method tested for the first- and second-order plants and numerical evaluation in terms of settling time, percent overshoot, steady-state error, rise time, integral of absolute error (IAE) and integral of squared error (ISE). The trajectories of the PID coefficients K_P, K_I and K_D are also analyzed to further asses the responsiveness, robustness and stability of the tuning methods. Additionally, the computational efficiency of the KAN-based tuner is compared to that of the MLP-based one.

All approaches were tested in the following set-ups: first-and second-plants with varying only noise conditions and for different step-input values; first-and second-plants varying both noise conditions and plant parameters under different step-input values; first- and second-order plants with fixed noise levels and plant parameters, but subjected to a time-varying input reference.

 doi:10.20944/preprints202604.2157.v1

3. Results

The KAN-based PID tuner developed in this work was evaluated against all comparative methods introduced in Section 2: the MLP-based tuner, the LMS adaptive tuner, the gain-scheduling controller, and the classical fixed-gain PID. Results are presented for three different testing setups, each designed to asses controller performance under a different source of difficulty- varying levels of input and output noise, changing plant parameters and variable reference complexity- step inputs and time-varying references for both first- and second-order linear dynamic systems.

3.1. Stochastic Disturbance: Varying Levels of Input and Output Noise

This subsection presents the results obtained for the first-order linear system cases, described in the Methods section, subjected to varying levels of input/ output noise. As a benchmark for further comparison and method efficiency evaluation we have included tests with no noise.

Figure 5 shows plot comparison results for controllers, employing all tested tuning methods, under zero noise for changing values of step inputs for first-order plants. All controllers manage to settle to the desired output. Gain scheduling performs best, achieving the fastest responses and lowest error metrics, as the values tested correspond directly to pre-set gains from its lookup table, it's closely followed by the two NN-based controllers- MLP and KAN and the LMS controller shows the most unsatisfactory performance of all, especially during large reference changes, due to its sensitivity to proper learning rate selection. Table 4 presents the numerical results of this comparison and further supports the conclusions drawn from the extracted plots.

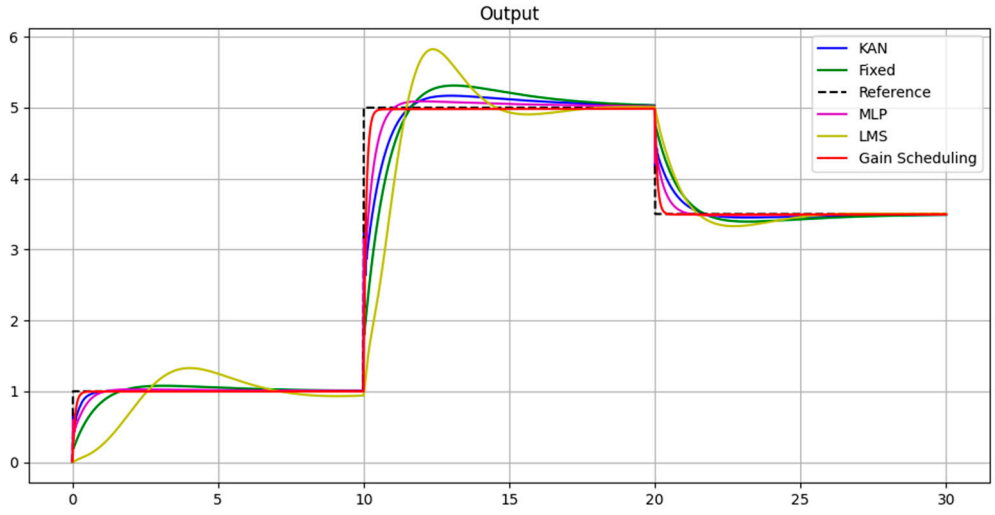


Figure 5. Outputs of all compared methods under zero noise conditions for first-order plant with parameters plant gain $K = 3$ and time constant $\tau = 15.0$. Simulation schedule: 0-10s: step input with value 1.0, input and output noise level equal to 0%, Plant gain $K = 3.0$, and time constant $\tau = 15.0$; 10-20s: step input with value 5.0, input and output noise level equal to 0%, Plant gain $K = 1.0$, and time constant $\tau = 10.0$; 20-30s: step input with value 3.5, input and output noise level equal to 0%, Plant gain $K = 3.0$, and time constant $\tau = 15.0$.

Figure 6 represents the time evolution of the proportional, integral and derivative gains generated by all compared tuning methods during the no-noise simulation. The KAN- and MLP-based tuners exhibit rapid gain adjustment during the initial transient stage and then settle to stable gain values once the desired system response is achieved, while the LMS controller continuously updates gains throughout the whole simulation course, which reflects its gradient-based nature reacting directly to the instantaneous tracking error. The gain scheduling controller maintains stable gains as there's only change in the reference magnitude, which is not considered a gain-scheduling event.

 doi:10.20944/preprints202604.2157.v1

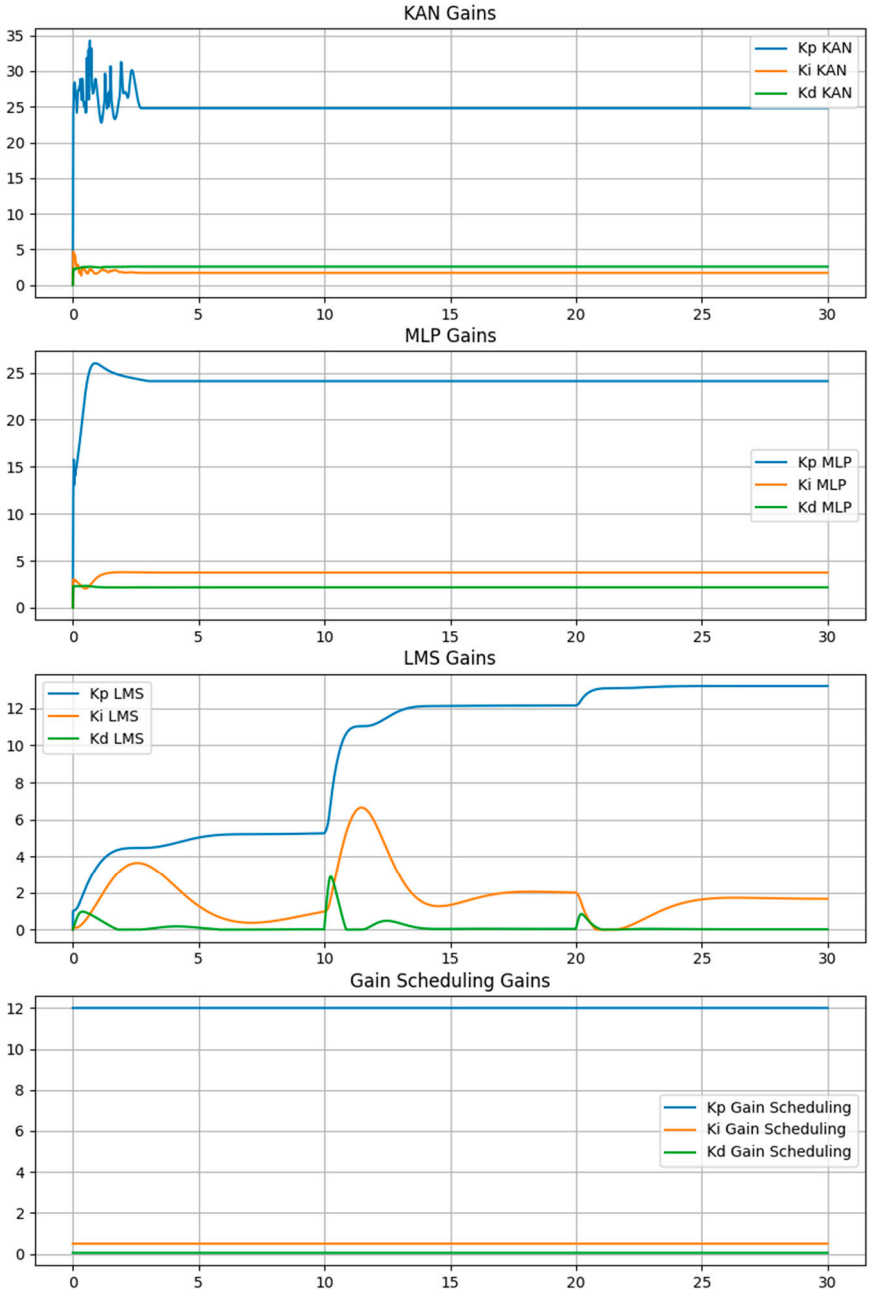


Figure 6. PID gains generated by all compared methods under zero noise conditions for first-order plant with parameters plant gain $K = 3$ and time constant $\tau = 15.0$. Simulation schedule: 0-10s: step input with value 1.0, input and output noise level equal to 0%, Plant gain $K = 3.0$, and time constant $\tau = 15.0$; 10-20s: step input with value 5.0, input and output noise level equal to 0%, Plant gain $K = 1.0$, and time constant $\tau = 10.0$; 20-30s: step input with value 3.5, input and output noise level equal to 0%, Plant gain $K = 3.0$, and time constant $\tau = 15.0$.

Table 4. Numerical comparison of LMS, MLP, KAN and gain scheduling tuning methods accuracy for first-order plants under zero noise conditions.

Metric	Time Interval	LMS	MLP	KAN	Gain Scheduling
Settling Time	0-10s	NaN	0.740494	0.920614	0.268000
	10-20s	13.884628	10.887258	14.449633	10.271000
	20-30s	24.468156	20.993996	21.154103	20.252000

 doi:10.20944/preprints202604.2157.v1

Percent Overshoot (%)	0-10s	32.716807	1.511340	3.191398	0.314162
	10-20s	20.393891	3.497833	20.393891	0.397377
	20-30s	11.544681	1.193212	4.433343	0.524088
Rise Time	0-10s	1.830610	0.560374	0.720480	0.194000
	10-20s	1.180393	0.580387	0.680454	0.195000
	20-30s	1.080360	0.640427	0.740494	0.186000
Steady-State Error	0-10s	0.060387	0.004780	0.006914	0.003104
	10-20s	0.007002	0.017468	0.020924	0.014922
	20-30s	0.000749	0.002432	0.005247	0.005030
IAE	0-10s	2.512062	0.270425	0.409768	0.123660
	10-20s	4.466569	0.969378	1.296082	0.509147
	20-30s	1.263058	0.309281	0.443723	0.175777
ISE	0-10s	1.347437	0.064266	0.090996	0.042839
	10-20s	8.447779	0.843588	1.053982	0.624790
	20-30s	0.772039	0.126646	0.159022	0.083870

In all experiments, noise was added to both the actuator signal and sensor output to replicate real industrial conditions. Three representative levels were tested: low (≈1% FS), medium (≈3% FS), and high noise (≈15% FS). Noise levels were varied dynamically during simulation to evaluate robustness under changing disturbance intensity. Figure 7 shows the plot comparison for this set-up and numerical results can be seen in Table 5.

Table 5. Numerical comparison of all tuning methods accuracy for first-order plants with simulation parameters as described in Figure 7.

Metric	Time Interval	LMS	MLP	KAN	Gain Scheduling
Settling Time	0-10s	NaN	0.920614	0.560374	0.296000
	10-20s	18.786262	19.753169	17.891928	10.297000
	20-30s	NaN	29.959973	NaN	NaN
Percent Overshoot (%)	0-10s	33.527145	3.815187	2.554122	3.379105
	10-20s	27.263110	53.890907	25.968467	2.841941
	20-30s	191.492182	20.786694	11.716738	43.480934
Rise Time	0-10s	1.700567	0.720480	0.420280	0.182000
	10-20s	1.220407	0.900600	1.200801	0.189000
	20-30s	0.220073	0.740494	1.260841	0.111000
Steady-State Error	0-10s	0.161102	0.001909	0.000511	0.014440
	10-20s	0.174716	0.075430	0.028167	0.013065
	20-30s	0.594151	0.017202	0.100775	0.379659
IAE	0-10s	2.858439	0.377145	0.180479	0.170249
	10-20s	5.231587	7.520717	4.090387	0.604415
	20-30s	7.100254	1.189996	2.562079	1.296958
ISE	0-10s	1.494917	0.085523	0.036996	0.043959
	10-20s	8.887588	11.141097	4.247738	0.633860
	20-30s	7.621526	0.378321	1.286500	0.314075

 doi:10.20944/preprints202604.2157.v1

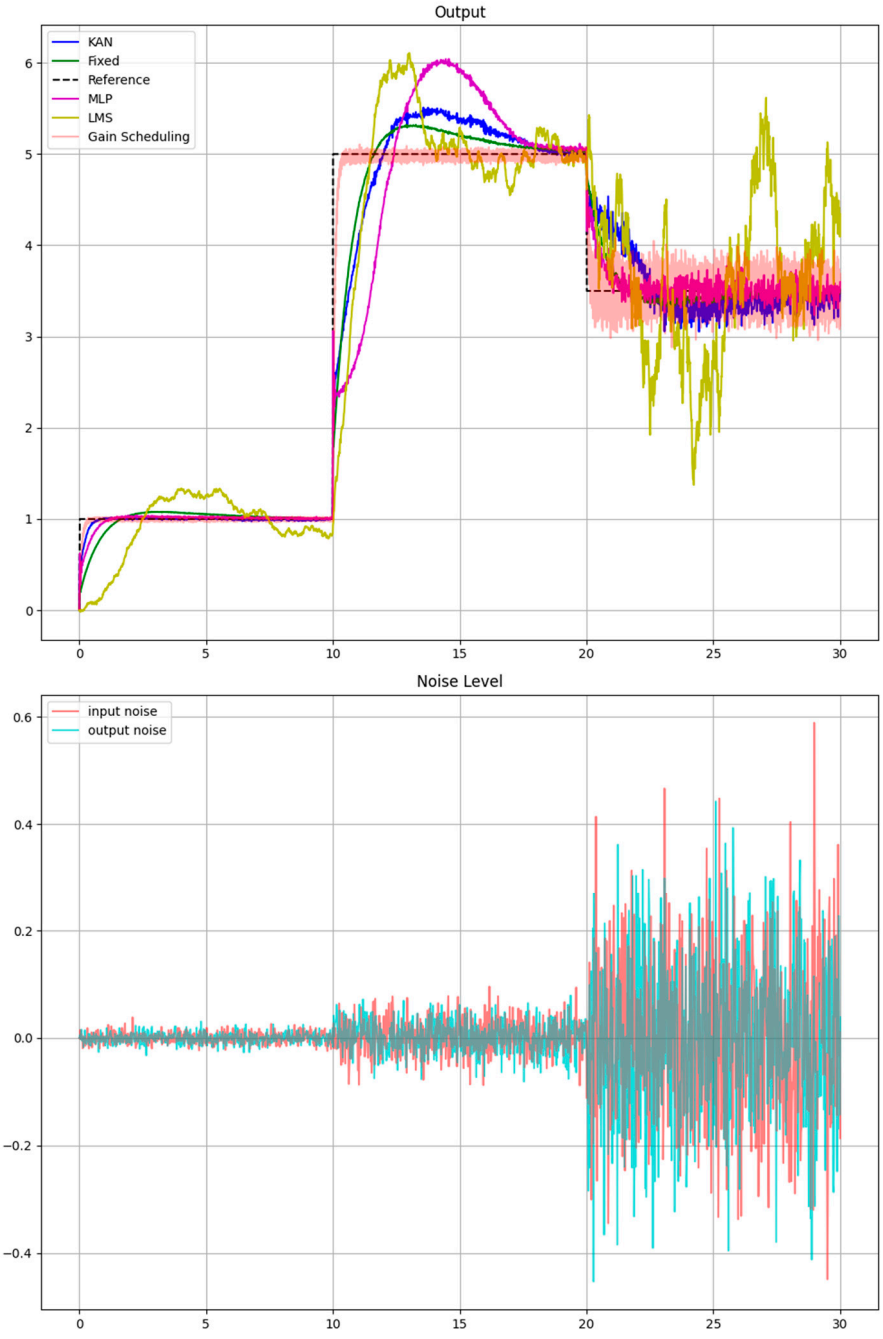


Figure 7. Results for first-order plant testing simulation with parameters: 0s-10s: step input with value 1.0, input noise level equal to 1%, output noise level equal to 1%, Plant gain K = 1.0, and time constant τ = 10.0; 10s-20s: step input with value 5.0, input noise level equal to 3%, output noise level equal to 3%, Plant gain K = 1.0, and time constant τ = 10.0; 20s-30s: step input with value 3.5, input noise level equal to 15%, output noise level equal to 15%, Plant gain K = 1.0, and time constant τ = 10.0.

The results show that noise has a significant impact on both transient behaviour and accumulated tracking error. KANs outperform MLPs, fixed PID coefficients and LMS in those testing conditions, demonstrating the highest robustness to noise. Under moderate and high noise levels both MLPs and LMS provide unstable diverging results, while KANs manage to settle. LMS performs poorly even with low noise levels, failing to settle in the ten second time interval of each test segment. Gain scheduling is again used as a benchmark for other methods, as the values tested have direct counterparts in its lookup table, but its performance also severely deteriorates under higher noise levels.

 doi:10.20944/preprints202604.2157.v1

A plot of the PID coefficients evolution throughout the simulation of the first-order plant subjected to different noise levels for the tested methods is presented in Figure 8. It is visible that the adaptive methods respond distinctly to the changing reference levels and noise intensities with KANs and MLPs providing strong adaptability, but accompanied by high gain variability, especially under higher noise levels, whereas the LMS tuner changes gains more conservatively, but provides unsatisfactory results. The gain scheduling serves as a baseline, as in those conditions no coefficient changes are performed.

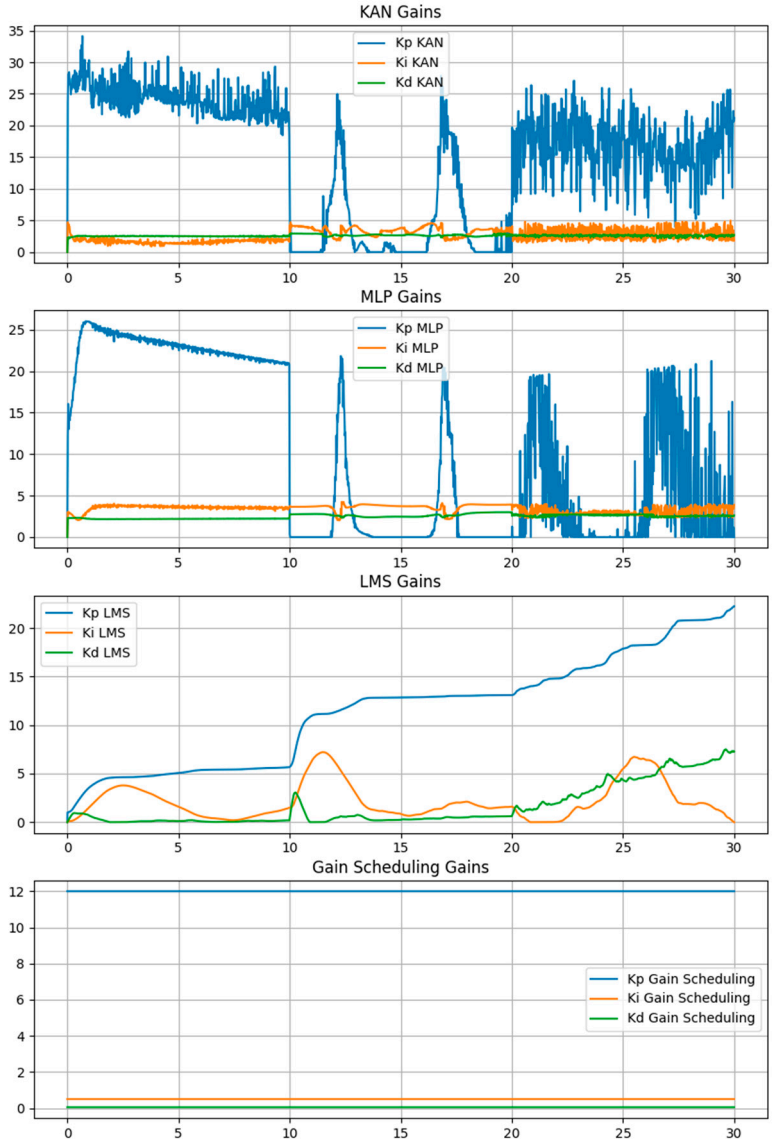


Figure 8. PID gains generated by all tested methods for first-order plant testing simulation with parameters: 0s-10s: step input with value 1.0, input noise level equal to 1%, output noise level equal to 1%, Plant gain K = 1.0, and time constant τ = 10.0; 10s-20s: step input with value 5.0, input noise level equal to 3%, output noise level equal to 3%, Plant gain K = 1.0, and time constant τ = 10.0; 20s-30s: step input with value 3.5, input noise level equal to 15%, output noise level equal to 15%, Plant gain K = 1.0, and time constant τ = 10.0.

3.2. Parametric Disturbance: Changing Plant Parameters

Testing for adaptation to plant parameter changes was done for both first- and second-order plants. Simulations are set up as in the previous subsection. Figure 7 shows plot comparison results comparing stable gain scheduling controller for first-order plants to controllers of all other discussed

 doi:10.20944/preprints202604.2157.v1

methods, while Figure 8 showcases gain scheduling's inability to adapt to unpredicted values. Numerical results are presented in Table 6 and Table 7.

Table 6. Numerical comparison of LMS, MLP, KAN and Gain Scheduling tuning methods performance for first-order plants with simulation parameters as described in Figure 6.

Metric	Time Interval	LMS	MLP	KAN	Gain Scheduling
Settling Time	0-10s	NaN	1.160774	6.464310	9.996000
	10-20s	NaN	10.927285	10.987325	10.401000
	20-30s	NaN	20.973983	21.054036	29.993000
Percent Overshoot (%)	0-10s	54.220626	4.972003	5.298795	10.919873
	10-20s	31.317646	2.035754	1.926445	2.672958
	20-30s	39.419905	10.332823	16.225244	26.519685
Rise Time	0-10s	5.371791	0.680454	0.470067	0.088000
	10-20s	4.541514	0.700467	0.740494	0.213000
	20-30s	1.080360	0.800534	1.100734	0.155000
Steady-State Error	0-10s	0.284945	0.003895	0.004328	0.031603
	10-20s	0.255966	0.023740	0.026004	0.044869
	20-30s	0.151371	0.018255	0.003690	0.102476
IAE	0-10s	5.383484	0.405777	0.470067	0.282226
	10-20s	11.937405	1.309222	1.290959	0.910913
	20-30s	10.194699	0.815737	0.919109	0.815465
ISE	0-10s	4.098881	0.095682	0.115444	0.031034
	10-20s	21.578159	1.600046	1.498544	0.831769
	20-30s	18.912526	0.424881	0.509843	0.2518872

Table 7. Numerical comparison of LMS, MLP, KAN and Gain Scheduling tuning methods performance for first-order plants with simulation parameters as described in Figure 7.

Metric	Time Interval	LMS	MLP	KAN	Gain Scheduling
Settling Time	0-10s	NaN	2.946631	2.366244	3.824000
	10-20s	19.656552	15.950634	15.950634	NaN
	20-30s	29.979993	25.316878	25.537025	20.224000
Percent Overshoot (%)	0-10s	27.129936	4.976931	6.488460	63.092734
	10-20s	9.915909	4.632208	4,622462	20.665626
	20-30s	39.769588	17.344877	19.155910	8.858477
Rise Time	0-10s	0.870290	0.620414	0.620414	0.032000
	10-20s	2.910970	0.740494	0.780520	0.167000
	20-30s	0.350117	0.900600	1.160774	0.175000
Steady-State Error	0-10s	0.042545	0.005727	0.006328	0.004651
	10-20s	0.043121	0.047300	0.052214	0.559866
	20-30s	0.008916	0.032570	0.030701	0.037007
IAE	0-10s	0.888357	0.345274	0.379568	0.255425
	10-20s	1.100741	2.015735	2.018902	6.147597
	20-30s	1.025537	1.394015	1.553468	0.544858
ISE	0-10s	0.267010	0.068368	0.067051	0.098387
	10-20s	0.383114	2.090578	1.991367	5.641962
	20-30s	0.178286	0.508392	0.607150	0.170250

 doi:10.20944/preprints202604.2157.v1

In these setups both stochastic disturbances and dynamic changes have been introduced simultaneously. Both Figures 9 and 11 and Tables 6 and 7 demonstrate how gain scheduling deteriorates when faced with unknown plant parameters, while learning-based controllers manage to maintain stable performance. KAN shows improved robustness under more severe conditions, followed closely by MLPs, and LMS fails to provide satisfactory results, especially in the testing conditions of Figure 7 and Table 6.

Gain evolution for those cases – Figures 10 and 12 is also similar, KANs show great variability under higher noise levels, while MLPs seem to provide constant values and LMS performs in an almost identical way to previous simulated cases showing error driven gain adaptation, due to its gradient-based nature.

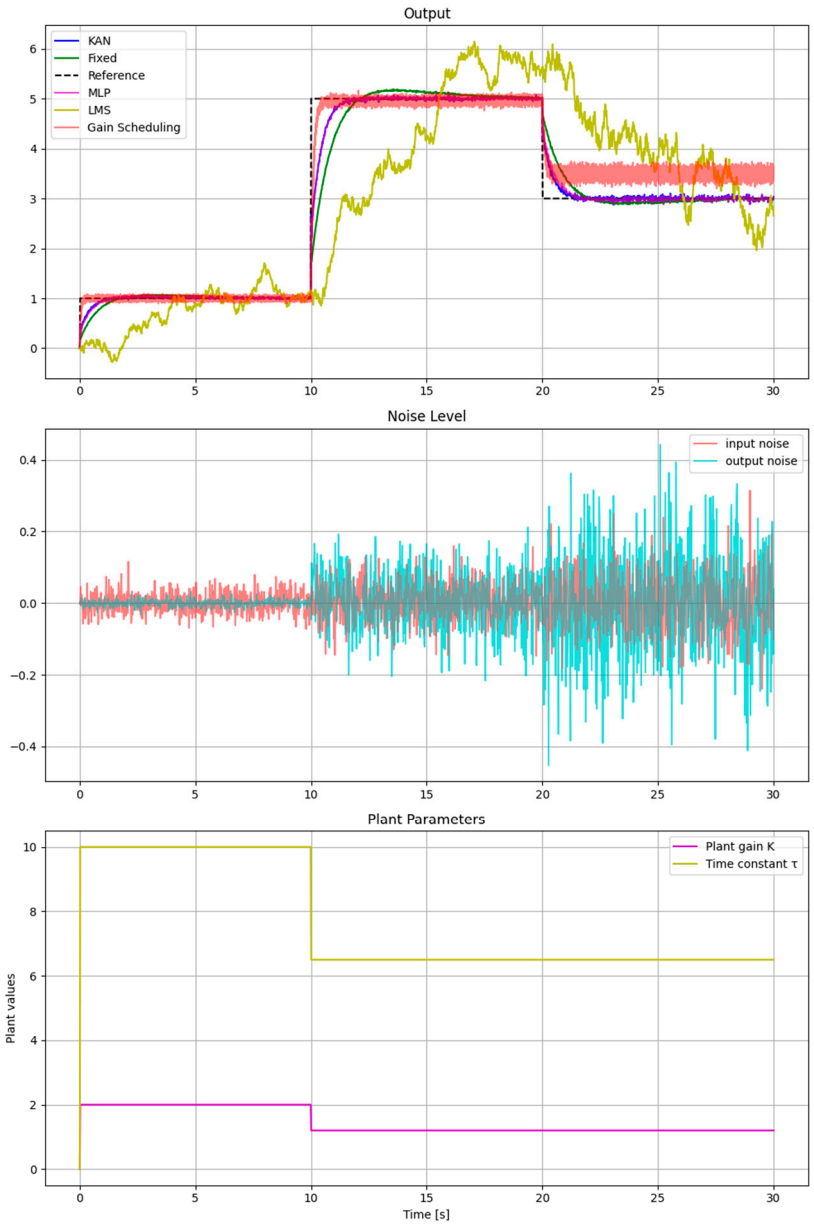


Figure 9. Results for first-order plant tested under the following simulation conditions: 0s-10s: step input with value 1.0, input noise level equal to 1%, output noise level equal to 3%, Plant gain K = 2.0, and time constant τ = 10.0; 10s-20s: step input with value 5.0, input noise level equal to 8%, output noise level equal to 5%, Plant gain K = 1.2, and time constant τ = 6.5; 20s-30s: step input with value 3.0, input noise level equal to 15%, output noise level equal to 8%, Plant gain K = 1.2, and time constant τ = 6.5.

 doi:10.20944/preprints202604.2157.v1

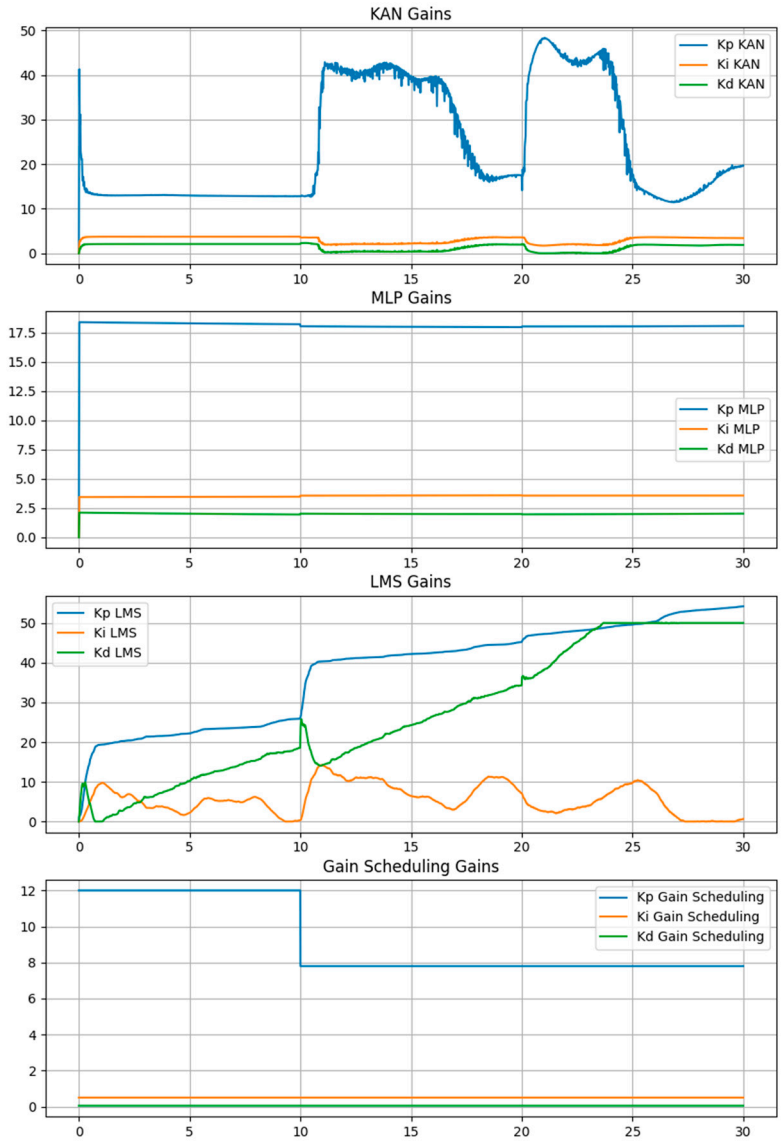


Figure 10. PID gains generated by all tested methods for first-order plant tested under the following simulation conditions: 0s-10s: step input with value 1.0, input noise level equal to 1%, output noise level equal to 3%, Plant gain K = 2.0, and time constant τ = 10.0; 10s-20s: step input with value 5.0, input noise level equal to 8%, output noise level equal to 5%, Plant gain K = 1.2, and time constant τ = 6.5; 20s-30s: step input with value 3.0, input noise level equal to 15%, output noise level equal to 8%, Plant gain K = 1.2, and time constant τ = 6.5.

 doi:10.20944/preprints202604.2157.v1

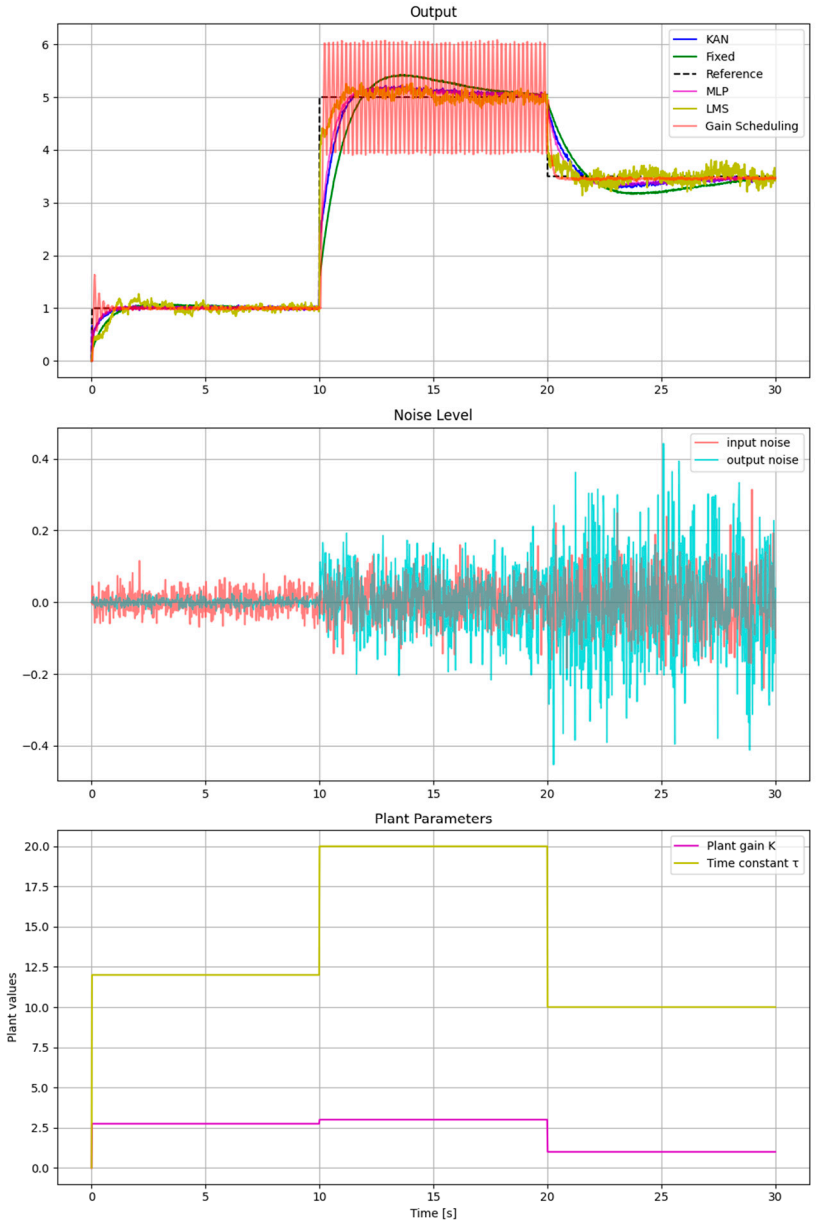


Figure 11. Results for simulation with parameters: 0s-10s: step input with value 1.0, input noise level equal to 1%, output noise level equal to 3%, Plant gain K = 2.75, and time constant τ = 12.0; 10s-20s: step input with value 5.0, input noise level equal to 8%, output noise level equal to 5%, Plant gain K = 3.0, and time constant τ = 20.0; 20s-30s: step input with value 3.5, input noise level equal to 15%, output noise level equal to 8%, Plant gain K = 1.0, and time constant τ = 10.0.

 doi:10.20944/preprints202604.2157.v1

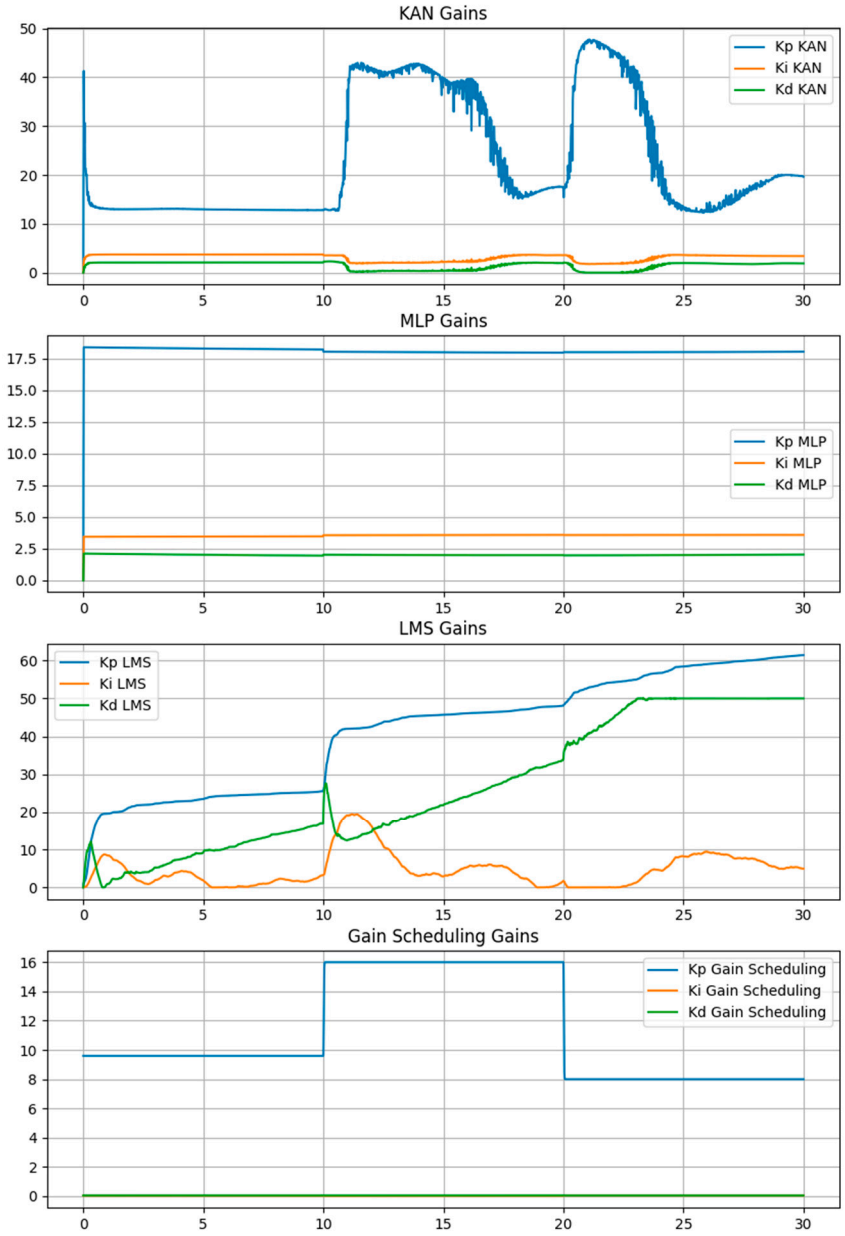


Figure 12. PID gains generated by all tested methods for first-order plant for simulation with parameters: 0s-10s: step input with value 1.0, input noise level equal to 1%, output noise level equal to 3%, Plant gain K = 2.75, and time constant τ = 12.0; 10s-20s: step input with value 5.0, input noise level equal to 8%, output noise level equal to 5%, Plant gain K = 3.0, and time constant τ = 20.0; 20s-30s: step input with value 3.5, input noise level equal to 15%, output noise level equal to 8%, Plant gain K = 1.0, and time constant τ = 10.0.

Figure 13 and Table 8 show performance comparison of all tuning methods for a second-order plant, subjected to reference changes, increasing input/output noise levels and variations in plant parameter- natural frequency and damping ratio.

 doi:10.20944/preprints202604.2157.v1

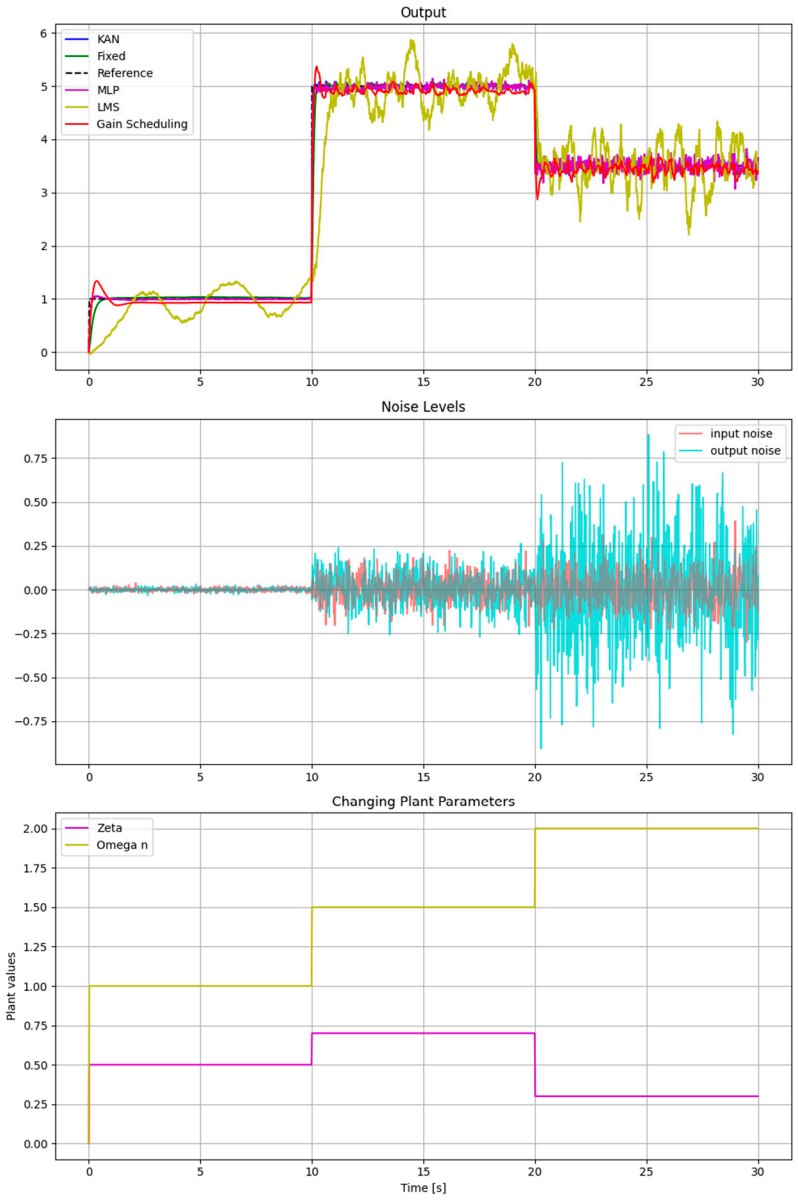


Figure 13. Results for second- order plants, subjected to the following testing schedule: 0s-10s: step input with value of 1.0, input noise level - 1%, output noise level - 1%, natural frequency $\omega_n = 1.0$, Damping ratio $\zeta = 0.5$; 10s-20s: step input with value of 5.0, input noise level - 10%, output noise level - 7%, natural frequency $\omega_n = 1.5$, damping ratio $\zeta = 0.7$; 20s-30s: step input with value of 3.5, input noise level - 30%, output noise level - 10%, natural frequency $\omega_n = 2.0$, damping ratio $\zeta = 0.3$.

Table 8. Numerical comparison of all tuning methods performance for second-order plants with simulation parameters as described in Figure 8.

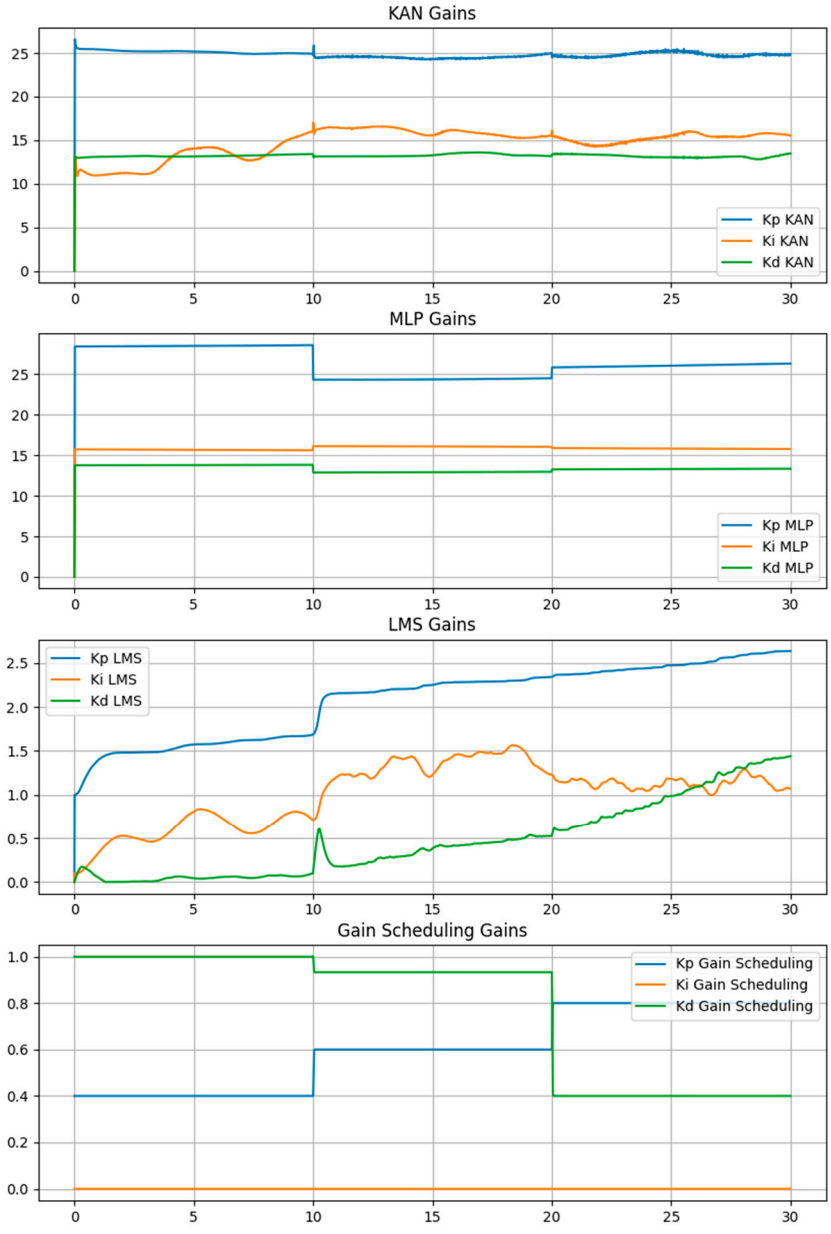
Metric	Time Interval	LMS	MLP	KAN	Gain Scheduling
Settling Time	0-10s	NaN	0.400267	0.400267	NaN
	10-20s	NaN	19.753169	19.753169	19.733155
	20-30s	29.979993	NaN	29.979987	NaN
Percent Overshoot (%)	0-10s	39.892829	5.076550	5.251044	33.659878
	10-20s	24.060029	8.825302	8.996116	14.307039
	20-30s	22.400868	9.063412	14.022039	17.959804
Rise Time	0-10s	1.450483	0.100067	0.120080	0.120080
	10-20s	0.590197	0.040027	0.040027	0.080053
	20-30s	0.130043	0.060040	0.060040	0.020013
Steady-State Error	0-10s	0.379553	0.002351	0.007223	0.064470

 doi:10.20944/preprints202604.2157.v1

	10-20s	0.337301	0.012777	0.012777	0.081759
	20-30s	0.064031	0.011901	0.006385	0.156504
IAE	0-10s	2.829726	0.104293	0.122321	0.851932
	10-20s	4.211281	0.447179	0.447729	1.085951
	20-30s	3.341427	0.876517	0.866168	0.918527
ISE	0-10s	1.338234	0.020738	0.022472	0.110818
	10-20s	5.673532	0.064042	0.059072	0.275897
	20-30s	1.834186	0.118545	0.115195	0.153222

For second-order systems, which introduce oscillatory behaviour and make achieving satisfactory controller performance more challenging, the results again demonstrate that learning-based controllers significantly outperform LMS and Gain scheduling. The difference is most noticeable under moderate and severe noise conditions. Both MLPs and KANs achieve nearly identical performance, with a slight advantage for KANs under the most severe conditions.

As seen in Figure 14, both KANs and MLPs manage to establish quite stable gain values throughout this simulation with MLPs providing superior performance, unlike LMS where gain values remain dynamic. Gain scheduling serves as a baseline for gain- switching events.



 doi:10.20944/preprints202604.2157.v1

Figure 14. PID gains generated by all tested methods for second- order plants, subjected to the following testing schedule: 0s-10s: step input with value of 1.0, input noise level - 1%, output noise level - 1%, natural frequency $\omega_n = 1.0$, Damping ratio $\zeta = 0.5$; 10s-20s: step input with value of 5.0, input noise level - 10%, output noise level - 7%, natural frequency $\omega_n = 1.5$, damping ratio $\zeta = 0.7$; 20s-30s: step input with value of 3.5, input noise level - 30%, output noise level - 10%, natural frequency $\omega_n = 2.0$, damping ratio $\zeta = 0.3$.

3.3. Reference Complexity: Step Inputs Versus Smoothly Time-Varying References

As the main focus of our study is the performance of the developed KAN PID tuner we first tested its ability to output different functions individually and compared it with the MLP-based tuner – Figure 15 and Figure 16.

Both tests show the learning-based tuners are able to deal with references more complex than just step inputs. Next simulations compare the controllers' ability to track time-varying reference signals.

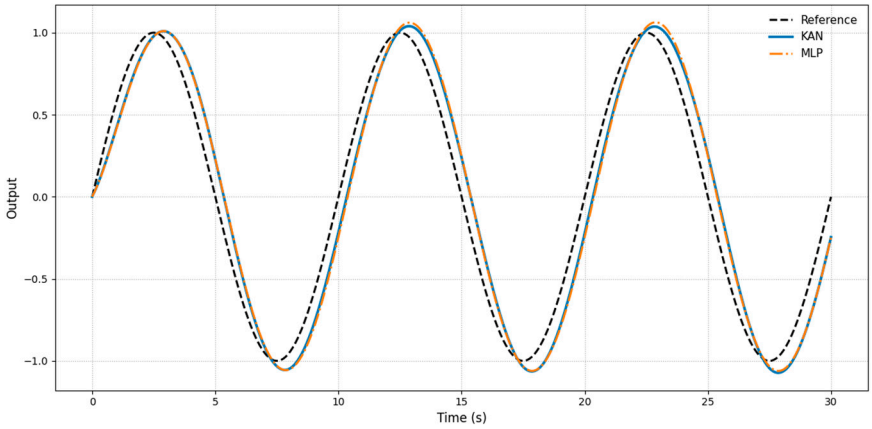


Figure 15. KAN- vs. MLP-based tuner outputs for sine function reference with amplitude = 1 and frequency = 0.1 for a system with a first-order plant.

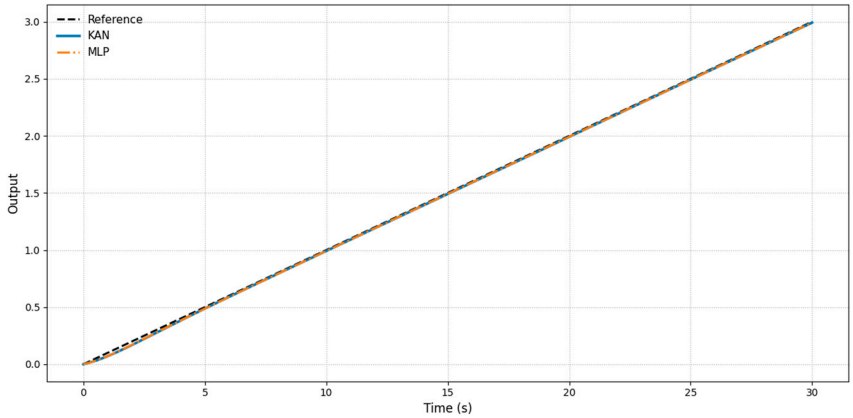


Figure 16. KAN- vs. MLP-based tuner outputs for ramp function reference for a system with a first-order plant.

First- and second-order plant systems, tuned with KANs, LMS, MLPs and gain scheduling are compared. Figure 17 shows plot comparison of all tested methods for systems with first-order plants, while Figure 19 shows the same experimental setup but for second-order plant systems.

 doi:10.20944/preprints202604.2157.v1

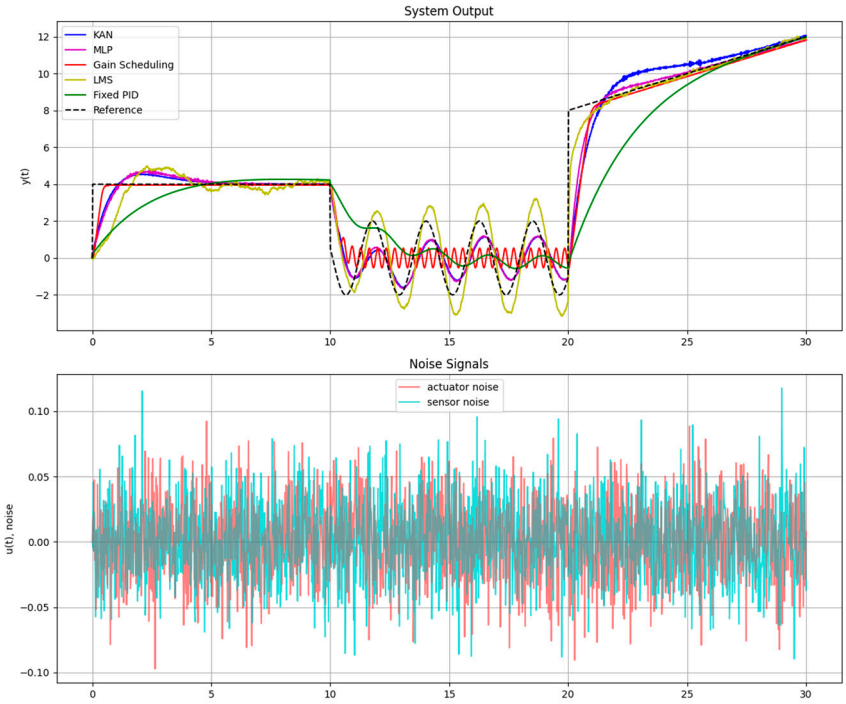


Figure 17. Simulation results for first-order plant systems with the following schedule: 0s-10s: step input with value 4.0, 10s-20s: sine input with amplitude 4.0 and frequency of 2.8; 20s-30s: ramp input with a constant slope of 0.4.

In Figure 17 the first-order plant is subjected to a sequence of references- step input, followed by a sinusoidal reference and a ramp input. KANs and MLPs exhibit nearly identical tracking performance, managing to achieve high accuracy in sections one and three with fast convergence and minimal steady-state error. The LMS controller I able to also provide satisfactory results, but suffers more from the introduced noise. Gain scheduling fails in the sinusoidal section and fixed PID coefficients are only optimal in the initial section.

All adaptive methods show great gain adjustments in Figure 18, providing stable gain values. MLPs tend to overshoot the proportional component as references are switched, but manage to settle nicely after that.

Figure 19 shows simulation results for second-order plants, subjected to a reference schedule of a ramp input, followed by a sinusoidal reference and a step input. Here the difference between the learning- and non-learning- based methods becomes even more apparent. KANs and MLPs exhibit perfect tracking, with KANs slightly outperforming in the sinusoidal section, demonstrating improved robustness. LMS performance deteriorates in the step section introducing significant overshoot and oscillations before settling ang Gain scheduling again performs poorly.

 doi:10.20944/preprints202604.2157.v1

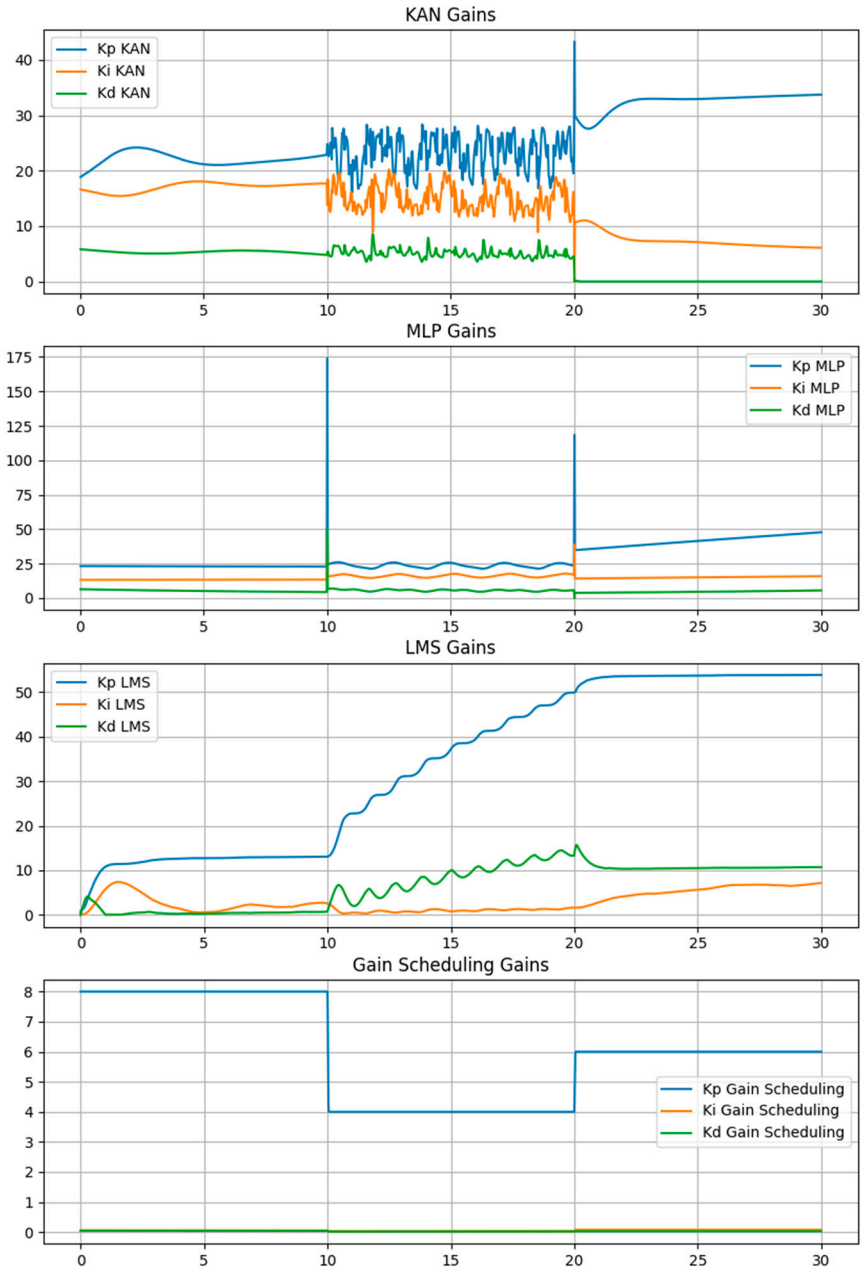


Figure 18. PID gains generated by all tested methods for simulation results for first-order plant systems with the following schedule: 0s-10s: step input with value 4.0, 10s-20s: sine input with amplitude 4.0 and frequency of 2.8; 20s-30s: ramp input with a constant slope of 0.4.

 doi:10.20944/preprints202604.2157.v1

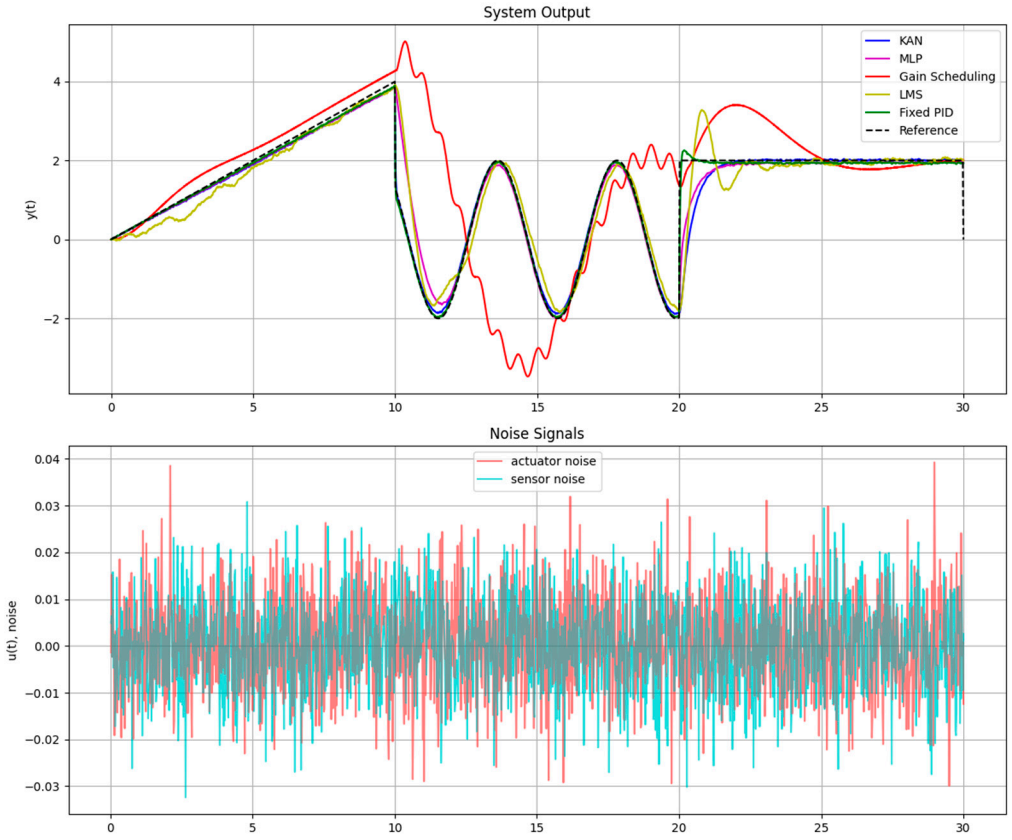


Figure 19. Simulation results for second-order plant systems with the following schedule: 0s-10s: , ramp input with a constant slope of 0.4; 10s-20s: sine input with amplitude 2.0 and frequency of 1.5; 20s-30s: step input with value 1.0.

Figure 20 demonstrates the methods' ability to adapt PID gains to the changing reference signals. The KAN and MLP controllers provide smooth gain adaptation across the three types of inputs, while the LMS-based tuner again shows gains, based on the accumulated tracking error and gain scheduling remains limited to its predefined values.

KANs consistently achieve accurate reference tracking while requiring less complete training datasets, showing improved data efficiency and generalization capabilities. While MLPs remain highly effective, their performance is more sensitive to training data richness, particularly in second-order systems. LMS, although computationally lightweight, is unable to cope effectively with rapid reference changes and higher-order dynamics.

 doi:10.20944/preprints202604.2157.v1

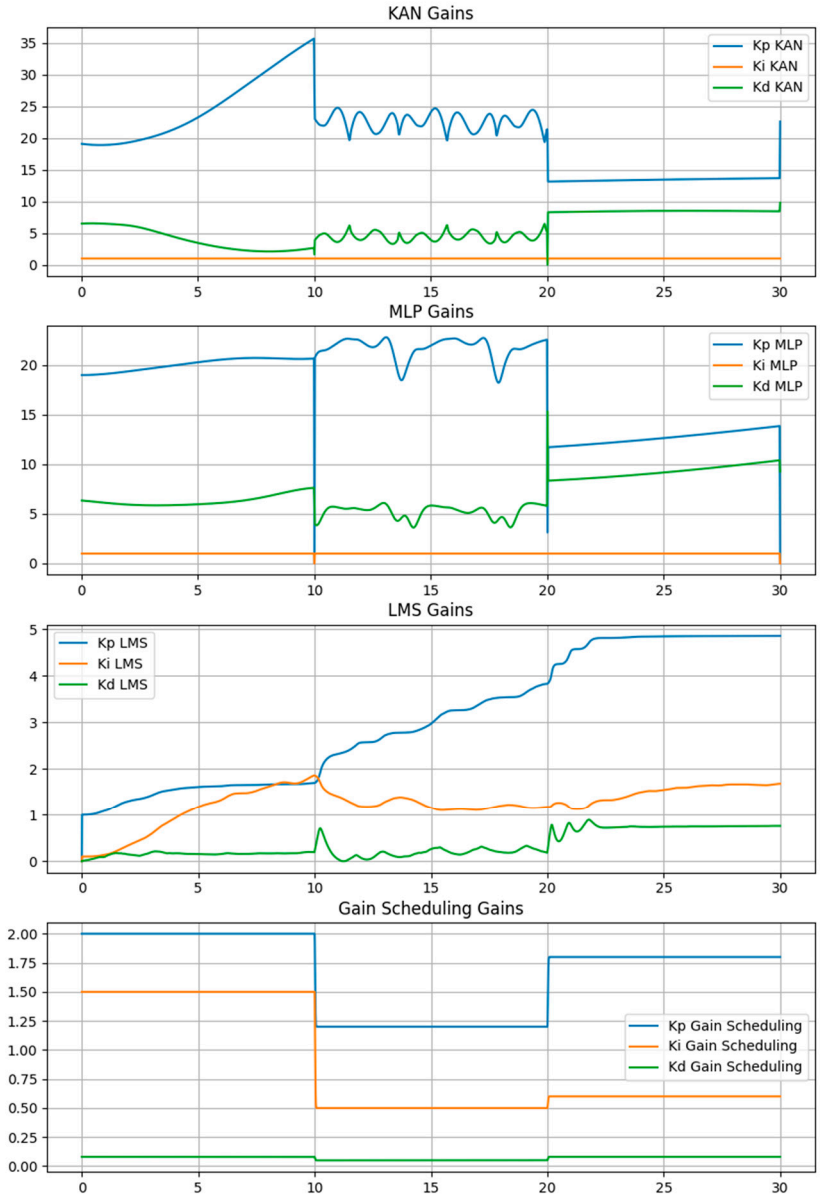


Figure 20. PID gains generated by all tested methods for Simulation results for second-order plant systems with the following schedule: 0s-10s: , ramp input with a constant slope of 0.4; 10s-20s: sine input with amplitude 2.0 and frequency of 1.5; 20s-30s: step input with value 1.0.

4. Discussion

The primary focus of this work is the development, evaluation and validation of a Kolmogorov-Arnold Network-based adaptive PID tuner for first- and second-order linear dynamic systems, operating under time-varying noise levels, plant parameters and reference trajectories. The remaining tuning approaches – MLPs, LMS and classical gain scheduling were inspired by other works and included as benchmark methods to compare the performance of the proposed KAN tuner [15,25,30].

4.1. Interpretation of Results

The simulation results consistently demonstrate that learning-based tuning approaches outperform classical gain scheduling and LMS methods. The difference is most visible under non-stationary operating conditions and tracking time-varying reference signals. The outcome of our

 doi:10.20944/preprints202604.2157.v1

study is in line with prior studies, which have indicated that fixed PID tuning strategies provide limited results when it comes to varying system dynamics, working conditions and input signals, as they are generally optimized for a narrow set of conditions [31,32].

Across all tested scenarios, the KAN-based tuner achieved superior or comparable accuracy, reduced steady-state error and lower integral error metrics when compared to other methods. The most noticeable improvement was observed during reference transitions involving sinusoidal and ramp inputs, where gain scheduling and LMS struggled to cope with the more complex and noisy dynamics. KANs exhibit overall improved adaptability compared to classical tuning methods.

For first-order linear plants both KANs and MLPs perform nearly identically when it comes to settling time, percent overshoot and tracking accuracy, but as system complexity increases in second-order plants KANs demonstrate improved robustness, particularly under highly noisy conditions and during reference transitions. This further supports the claim that KANs generalize better when handling complex scenarios and nonideal conditions as adaptive tuners.

The LMS-based controller showed limited adaptability even under moderate noise levels. Performance degraded rapidly when faced with rapidly changing references and high input/output noise levels. Its computational simplicity is not enough to justify the unsatisfactory performance. Those findings align with the know limitations of gradient-based adaptive methods, which lack the expressive power needed to model the dependencies between plant dynamics and optimal PID gains, and are highly sensitive to proper parameter selection.

4.2. Gain Adaptation Behaviour

The difference in how the methods, compared in our paper, respond to changing plant dynamics, noise levels and reference signals becomes apparent when the evolution of the PID gains throughout the tested scenarios is considered. The KAN-based controller consistently shows fast gain adaptation, mainly through it proportional and derivative components responding to shifts in reference type and magnitude, changes in noise levels and plant parameters. While this provides great output adaptation, it also introduces lots of gain variability, especially under higher noise levels. MLPs behave differently, producing smoother and more constant gain profiles, but this can limit their adaptability in highly changeable environments. LMS gains are a direct reflection of the method's gradient-based nature, adapting gradually to the accumulated tracking error, and gain scheduling shows abrupt changes only at the instances of the occurrence of the detects events.

A trade-off between adaptability and gain variation is clearly observed- the learning-based methods are superior when it comes to responsiveness, the LMS tuner performs in a more stable and predictable manner, but outputs inferior performance, and gain scheduling lacks the flexibility the other tuners have.

4.3. Computational Efficiency and Model Complexity

Another angle explored in this work is the comparison of computational efficiency and model complexity between the similarly performing KAN and MLP tuners. Here KANs demonstrate superior performances with more compact models and less optimized training datasets.

As presented in Table 9, KAN models employed compact network configurations, while MLPs required substantially wider architectures to achieve similar performance.

Table 9. Comparison of network architectures used for KAN- and MLP-based adaptive PID tuners for first- and second-order plant systems.

Model Type	System Order	Network Architecture	Hyperparameters
KAN	First-order	[4, 4, 4, 3] – [5, 4, 4, 3]	grid = 3, k = 3
	Second-order	[4, 4, 4, 3] – [7, 4, 4, 3]	grid = 3, k = 3
MLP	First-order	[4, 64, 64, 32, 3]	-
	Second-order	[7, 64, 64, 32, 3]	-

 doi:10.20944/preprints202604.2157.v1

Throughout our testing KAN-based tuners consistently matched or outperformed MLPs, particularly in the second-order plant cases. The significantly smaller networks manage to efficiently approximate the nonlinear control mappings, highlighting one of KANs main practical advantages-model simplicity. KANs exhibit improved data efficiency and reduced memory requirements, making them an option worth exploring more for real-time control applications with limited computational resources. The results support the claims of the original KAN paper, which states that high approximation accuracy can be achieved without relying on excessive network width and depth, unlike in MLP-based solutions [23].

4.4. Broader Implications for Adaptive Control

Our findings reinforce the limitations of fixed PID coefficients and gain scheduling approaches under rapidly changing operating conditions. Both learning-based tuners, proposed in our paper, manage to demonstrate that this approach can effectively overcome the aforementioned limitations by continuously updating controller parameters.

KANs' superior performance in second-order plant systems is relevant for mechanical, electromechanical and process control applications, where oscillatory dynamics, fluctuating parameters and measurement noise are a common occurrence [33]. Other advantages of KANs are their inherent generalization abilities, model efficiency and potential interpretability, when compared to MLPs.

4.5. Limitations and Future Research Directions

While the results of our study are promising and support KANs effectiveness for control problems, the testing work was limited to first- and second-order linear plant models. Expanding the framework to higher order, nonlinear and time-delay systems should be the next step explored. Explicit measurements of runtime and energy consumption would also support the claims for improved computational efficiency, which was only inferred from network sizes.

Further research directions include: experimental validation on real-world physical systems, exploring online training strategies for KAN-based tuners, hybrid control schemes and more.

As a conclusion, our work explores KAN-based adaptive PID tuners as an effective, efficient and scalable alternative to both classical and neural-network-based tuning methods, particularly in uncertain control environments.

Author Contributions: Conceptualization, V.P. and S.A.-S.; methodology, L.C.; software, L.C.; validation, S.A.-S. and N.S.; formal analysis, V.P.; investigation, L.C.; resources, V.P. and ; data curation, L.C.; writing—original draft preparation, L.C.; writing—review and editing, V.P.; visualization, L.C.; supervision, S.A.-S.; project administration, N.S.; funding acquisition, N.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Regional Development Fund within the OP "Research, Innovation and Digitalization Programme for Intelligent Transformation 2021-2027", Project CoC "Smart Mechatronics, Eco- and Energy Saving Systems and Technologies", No. BG16RFPR002-1.014-0005.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: In this section, you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments). Where GenAI has been used for purposes such as generating text, data, or graphics, or for study design, data collection, analysis, or interpretation of data, please add "During the preparation of this manuscript/study, the author(s) used [tool name, version information] for the purposes of [description of use]. The authors have reviewed and edited the output and take full responsibility for the content of this publication."

 doi:10.20944/preprints202604.2157.v1

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

KAN	Kolmogorov-Arnold Network
MLP	Multilayer perceptron
PID	Proportional-integral-derivative
ReLU	Rectified Linear Unit
LBFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm
MSE	Mean Squared Error
LMS	Least mean squares
NN	Neural network
KAT	Kolmogorov-Arnold representation theorem
IAE	Integral of Absolute Error
ISE	Integral of Squared Error
FS	Full Scale

References

1. *Practical PID Control*; Visioli, A., Ed.; Advances in Industrial Control; Springer London: London, 2006; ISBN 978-1-84628-585-1.
2. Rana, K.P.S. Fuzzy Control of an Electrodynamic Shaker for Automotive and Aerospace Vibration Testing. *Expert Systems with Applications* **2011**, *38*, 11335–11346, doi:10.1016/j.eswa.2011.02.184.
3. Ulkir, O.; Ertugrul, I. Application of PID Controller in Chemical Process System Using COMSOL. *J. mechatron., artif. intell., eng.* **2022**, *3*, 107–113, doi:10.21595/jmai.2022.23035.
4. Kabir, U.; Hamza, M.F.; Haruna, A.; Shehu, G.S. Performance Analysis of PID, PD and Fuzzy Controllers for Position Control of 3-Dof Robot Manipulator 2019.
5. Pan, I.; Das, S. Fractional Order Fuzzy Control of Hybrid Power System with Renewable Generation Using Chaotic PSO. **2016**, doi:10.48550/ARXIV.1611.09809.
6. Naseer, O.; Khan, A.A. Hybrid Fuzzy Logic and Pid Controller Based Ph Neutralization Pilot Plant 2013.
7. Aamir, M. On Replacing PID Controller with ANN Controller for DC Motor Position Control. *IJRSC* **2013**, *2*, doi:10.5861/ijrsc.2013.236.
8. Hu, N. The Limitations of Traditional PID Controllers and Modern Optimization Methods. *ACE* **2025**, *147*, 238–244, doi:10.54254/2755-2721/2025.22912.
9. Åström, K.J.; Hägglund, T. *PID Controllers: Theory, Design, and Tuning*; 2. ed.; Instrument Society of America: Research Triangle Park, NC, 1995; ISBN 978-1-55617-516-9.
10. Åström, K.J.; Hägglund, T.; Hang, C.C.; Ho, W.K. Automatic Tuning and Adaptation for PID Controllers - a Survey. *Control Engineering Practice* **1993**, *1*, 699–714, doi:10.1016/0967-0661(93)91394-C.
11. Landau, I.D.; Zito, G. Digital Control Systems - New Edition (I. D. Landau & G. Zito). **2020**, doi:10.13140/RG.2.2.19321.49764.
12. Holland, J.H. Genetic Algorithms. *Sci Am* **1992**, *267*, 66–72, doi:10.1038/scientificamerican0792-66.
13. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the Proceedings of ICNN'95 - International Conference on Neural Networks; IEEE: Perth, WA, Australia, 1995; Vol. 4, pp. 1942–1948.
14. Narendra, K.S.; Parthasarathy, K. Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Trans. Neural Netw.* **1990**, *1*, 4–27, doi:10.1109/72.80202.
15. Almachi, J.C.; Vicente, R.; Bone, E.; Montenegro, J.; Cando, E.; Reina, S. Implementation of a Neural Network for Adaptive PID Tuning in a High-Temperature Thermal System. *Energies* **2025**, *18*, 3113, doi:10.3390/en18123113.
16. Lakhani, A.I.; Chowdhury, M.A.; Lu, Q. Stability-Preserving Automatic Tuning of PID Control with Reinforcement Learning. *Complex Eng Syst* **2022**, *2*, 3, doi:10.20517/ces.2021.15.

 doi:10.20944/preprints202604.2157.v1

17. Marino, A.; Neri, F. PID Tuning with Neural Networks. In *Intelligent Information and Database Systems*; Nguyen, N.T., Gaol, F.L., Hong, T.-P., Trawiński, B., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, 2019; Vol. 11431, pp. 476–487 ISBN 978-3-030-14798-3.
18. Vaca-Rubio, C.J.; Blanco, L.; Pereira, R.; Caus, M. Kolmogorov-Arnold Networks (KANs) for Time Series Analysis 2024.
19. Wang, S.; Luo, W.; Yin, S.; Zhang, J.; Liang, Z.; Zhu, Y.; Li, S. Interpretable State Estimation in Power Systems Based on the Kolmogorov–Arnold Networks. *Electronics* **2025**, *14*, 320, doi:10.3390/electronics14020320.
20. Sanfilippo, S.; Hernández-Gálvez, J.J.; Hernández-Cabrera, J.J.; Évora-Gómez, J.; Roncal-Andrés, O.; Caballero-Ramirez, M. Evolving Electricity Demand Modelling in Microgrids Using a Kolmogorov-Arnold Network. *Informatica* **2025**, 1–22, doi:10.15388/25-INFOR590.
21. Wang, Y.; Sun, J.; Bai, J.; Anitescu, C.; Eshaghi, M.S.; Zhuang, X.; Rabczuk, T.; Liu, Y. Kolmogorov–Arnold-Informed Neural Network: A Physics-Informed Deep Learning Framework for Solving Forward and Inverse Problems Based on Kolmogorov–Arnold Networks. *Computer Methods in Applied Mechanics and Engineering* **2025**, *433*, 117518, doi:10.1016/j.cma.2024.117518.
22. Andrei Nikolaevich Kolmogorov On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition. *Doklady Akademii Nauk* **1957**, *114*, 953–956.
23. Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljačić, M.; Hou, T.Y.; Tegmark, M. KAN: Kolmogorov-Arnold Networks 2025.
24. Demirtaş, M. A Hybrid Algorithm for Adaptive Neuro-Controllers. *Black Sea Journal of Engineering and Science* **2023**, *6*, 87–97, doi:10.34248/bsengineering.1238543.
25. Sami Hasan; Aya Moufak Ismael Implementable Self-Learning PID Controller Using Least Mean Square Adaptive Algorithm. *eijs* **2021**, 148–154, doi:10.24996/ijs.2021.SI.1.20.
26. Widrow, B.; Hoff, M.E. (1960) Bernard Widrow and Marcian E. Hoff, "Adaptive Switching Circuits," 1960 IRE WESCON Convention Record, New York: IRE, Pp. 96-104. In *Neurocomputing, Volume 1*; Anderson, J.A., Rosenfeld, E., Eds.; The MIT Press, 1988; pp. 126–134 ISBN 978-0-262-26713-7.
27. Haykin, S.S. *Adaptive Filter Theory*; Prentice Hall information and system sciences series; 3. ed.; Prentice Hall: Upper Saddle River, NJ, 1996; ISBN 978-0-13-322760-4.
28. Finn Haugen *Basic Dynamics and Control*; TechTeach, 2010; ISBN 978-82-91748-13-9.
29. Vasickaninova Anna; Bakosova Monika; Oravec Juraj; Meszaros Alajos Gain-Scheduled Control of Counter-Current Shell-and-Tube Heat Exchangers in Series. *Chemical Engineering Transactions* **2018**, *70*, 1399–1404, doi:10.3303/CET1870234.
30. Vega, S.; Vasquez-Guevara, M.; Camacho, O. A Comparison of Adaptive PID, Adaptive Dual-PID and Adaptive Fractional PID Controllers for a Nonlinear System with Variable Parameters: In Proceedings of the Proceedings of the 21st International Conference on Informatics in Control, Automation and Robotics; SCITEPRESS - Science and Technology Publications: Porto, Portugal, 2024; pp. 166–177.
31. Hu, N. The Limitations of Traditional PID Controllers and Modern Optimization Methods. *ACE* **2025**, *147*, 238–244, doi:10.54254/2755-2721/2025.22912.
32. Çelik, D.; Khosravi, N.; Khan, M.A.; Waseem, M.; Ahmed, H. Advancements in Nonlinear PID Controllers: A Comprehensive Review. *Computers and Electrical Engineering* **2026**, *129*, 110775, doi:10.1016/j.compeleceng.2025.110775.
33. Hu, X.; Tan, W.; Hou, G. Tuning of PID/PIDD2 Controllers for Second-Order Oscillatory Systems with Time Delays. *Electronics* **2023**, *12*, 3168, doi:10.3390/electronics12143168.