Nonlinear Model Reference Adaptive Control (NMRAC) for First Order Systems

Dalim Wahby
Université Côte d'Azur
I3S/CNRS
Sophia Antipolis, France
wahby@i3s.unice.fr

Alvaro Detailleur
IDSC
ETH Zürich
Zürich, Switzerland
adetailleur@student.ethz.ch

Guillaume Ducard
Université Côte d'Azur
I3S/CNRS
Sophia Antipolis, France
ducard@i3s.unice.fr

***Abstract*—...**

***Keywords*—Neural Networks, Model Reference Adaptive Control, Lyapunov Method, Nonlinear Control, Stability, Adaptive Control, Robotics**

I. Introduction

Model reference control is a technique that imposes a behavior on a system, by equating the the output of the model reference with the output of the controlled system. This way, the gains of a pre-defined controller structure can be found.

However, this only works if the system's dynamics are known exactly, which is often far from reality. If the system's dynamics are not fully known, the controller gains can be found adaptively by a technique called *model reference adaptive control* (MRAC). The adaptation law can be chosen in two ways, firstly a gradient descent-based update law as first proposed by Whitaker et al. [1] and applied in for example **wahby** [2], and secondly, a Lyapunov-based update law as proposed by Shackcloth et al. [3]. The latter enforces that in every update step, we have a stabilizing controller, which is not guaranteed when using the former.

Since the structure of the control law is pre-defined, a rich enough model has found. To this end, neural networks (NN) are an ideal choice, since they are considered to be universal approximators [4]. Since at least the 90s, research has been conducted on NN controllers [5], and it has been shown that they can learn a desired behavior, for example in **wahby** [6]–[8]. However, standard NNCs are nonlinear, through the use of nonlinear activation functions, which makes their makes the formal verification of stability more challenging. Often these controllers rely on a posteriori stability verification, ...

which can become non-trivial or is not enough if we require online, stable learning techniques.

To ensure that a system engages in the desired behavior, we need to assess its stability, meaning we need to mathematically analyze and verify if a system converges to the desired output, given the input signal. If this is the case, we say the system is stable. While classical control theory primarily deals with linear time-invariant (LTI) systems, following the superposition principle, real-world systems often exhibit nonlinear behavior. This extends the scope of traditional control theory to handle these more complex, nonlinear systems, which leads us to not be able to use classical stability concepts, such as gain and phase margins. Hence, we need to use more generalized methods, such as Lyapunov theory, which aims to quantify dissipative energy in the system. However, finding an appropriate Lyapunov function can be challenging. When using MRACs with a Lyapunov-based update law, we can impose a Lyapunov function and ensure that the weights satisfy stability conditions on this function. In the MRAC literature however, this is only done for adaptive weights that appear in the form of a linear combination with the error dynamics of the system, meaning the learned parameters are not inside the nonlinear function. Hence, in this work we aim to extend on the MRAC principle and define a nonlinear update rule for MRAC controllers, where the estimated parameters are in a function that does not satisfy the superposition principle.

To this end, we first investigate the performance of stable linear MRACs compared to existing, not necessarily stable learning schematics, such as a PIDNN and a self-tuning PID controller, applied to an inverted pendulum, called Pendubot. With our insights on developing stable linear MRACs, we go ahead and tackle the problem of stable nonlinear MRACs. For reasons of simplicity, we begin by defining an update law for a first-order system and subsequently propose a MIMO update law. Both newly proposed update laws will are validated through simulations.

II. Literature Review and Contributions

...

A. Related Work

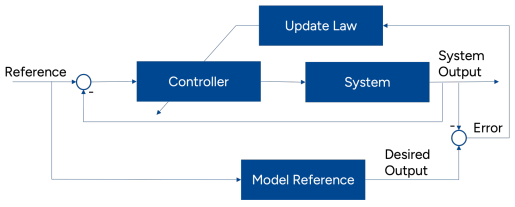


Figure 1: MRAC for first-oder systems controlled by simple NNCs

B. Contributions

...

III. Definitions and nomenclature

A. Systems and Stability

In order to define stable learning algorithms, it is imperal to firstly define, first-order systems and stability, which is taken from [9].

Definition 1 (System)**.** A continuous-time first-order system is defined by the following map

$$f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, (x(t), u(t)) \mapsto \dot{x}(t) = f(x(t), u(t)). \quad (1)$$

Additionally, if $u(t)$ is a function of x, the system is autonomous, with closed-loop dynamics $f(x(t))$. The trajectory of the system is defined by the evolution of $x(t)$. Furthermore, the system has an equilibrium point at $f(x(t)) = 0$.

Definition 2 (Stability)**.** Consider the equilibrium point $x = 0$ of (1). Then the system is...

- *stable*, if for each $\epsilon > 0$, there is a $\delta > 0$, sich that
$$||x(0)|| < \delta \Rightarrow ||x(t)|| < \epsilon, \quad \forall t \geq 0$$
- *unstable*, if not stable, and
- *asymptotically stable* if it is stable and δ can be chosen such that
$$||x(0)|| < \delta \Rightarrow \lim_{t \to \infty} x(t) = 0$$

From here henceworth, the dependency of the dynamics on time is considered to be implicit and will be neglected in the notation.

Definition 3 (Lyapunov function)**.** A Lyapunov function is a continuous function $V : \mathbb{R}^n \to \mathbb{R}$ with the following properties:

(a) positive definitness: $V(x) > 0, \quad \forall x \in \mathbb{R}^n \setminus \{0\}$ and $V(0) = 0$,
(b) decrease condition: $\dot{V}(x) \leq 0, \quad \forall x \in \mathbb{R}$.

Definition 4 (Stability in the sense of Lyapunov)**.** If there exists a continuous function $V : \mathbb{R}^n \to \mathbb{R}$ such that

(a) $V(x)$ is positive definite, and
(b) $\dot{V}(x) \leq -l(x)$, for some positive semidefinite function $l(x)$,

then the system is considered to be stable. Additionally, if $l(x)$ is positive definite, then the system is asymptotically stable.

The ultimate goal of this research is to develop stable update laws for neural network (NN) controllers. Hence, the designed control law will be considered as a NN. Hence, NNs are defined as follows throughout this work.

Definition 5 (Neural network)**.** A neural network (NN) $\phi : \mathbb{R}^n \to \mathbb{R}^p$ is defined as:

$$\begin{aligned} \phi(x) &= (L_1 \circ \varphi_1 \cdots \circ L_{H-1} \circ \varphi_{H-1} \circ L_H \circ \varphi_H)(x) \\ L_i(x) &= \theta_i x + b_i \quad \forall i \in \{1, ..., H\}, \end{aligned} \quad (2)$$

where $\varphi_i(\cdot)$ are called activation functions, θ_i and b_i are the weight matrix and bias of layer i, respectively.

Whenever, a bias is not mentioned it is assumed to be zero.

NNs usually make use of nonlinear activation functions, which enable them to approximate nonlinear functions. Typically, functions such as the hyperbolic tangent, sigmoid, or ReLU are used in machine learning. This work will utilize an actiavtion function that is designed to model a smooth saturation function, as used in **wahby**[7] and defined in (3). Note, that it activation function saturates at $\pm\frac{2}{a}$.

$$\sigma(x) = \frac{2(1 - e^{-ax})}{a(1 + e^{-ax})} \quad (3)$$

IV. Nonlinear Model Reference Adaptive Control (NMRAC)

One of the goal of this work is to analyse stability properties of nonlinear NNCs. To this end, we propose stable update scheme. We commence by looking at the simpler case of a first-order system. More, specifically, we define a problem, where we show that the even in the nonlinear case, we are able to learn a parameter with an algorithm that is based on the logic of a linear MRAC. We call this nonlinear Model Reference Adaptive Control (NMRAC).

We desire to control a system of the form, as described in equation (4), where $\phi : \mathbb{R} \to \mathbb{R}$ is a nonlinear function, $e_x = x_r - x$,and θ is the parameter we aim to learn. Here, we would like to point that θ can be seen as a weight, and ϕ the activation function of a NN of the form of a sigmoidial saturation function, as described in equation (3).

$$\dot{x} = -ax + b\phi(\theta e_x) \quad (4)$$

Next, we define a stable reference model, as defined in equation (5), where $e_m = x_r - x_m$, and $a_m > 0$. Note, that we want ϕ_m to be of the same form as ϕ.

$$\dot{x}_m = -a_m x_m + b_m \phi_m(\theta_m e_m) \quad (5)$$

Since we aim for the system to follow the reference model, we define the error dynamics as in equation (6). Note, that from this definition it follows that $e = e_x - e_m$. We will use this alternative definition later for the stability analysis of the update law.

$$e = x_m - x \quad (6)$$

Following the error dynamics, we define the Lyapunov candidate to be defined as in equation (7). The factor $c > 0$, is used to accelerate or decelerate the learning process. The norm of the error captures the distance between the internal states of the reference model and the system, and the term $||\alpha||_2^2$ captures the distance between the NNC and the desired NNC, which can be seen as the difference in dynamics between the controlled system and the model reference. Note, that both e and α should be 0, when our system follows the reference model and the desired parameters are learned. Futhermore,

this property renders our Lyapunov candidate to be positive definite.

$$V(e,\alpha) = ||e||_2^2 + ||c\alpha||_2^2 \tag{7}$$

To ensure that the controlled system is stable, we require the time derivative of the Lyapunov function to be negative. Therefore, we analyze the behavior of the resulting time derivative of the Lyapunov function. The resulting equation is defined in (8).

$$\dot{V}(e,\alpha) = 2e\dot{e} + 2\alpha\dot{\alpha}c \tag{8}$$

We now construct the time derivative of the error dynamics, which is defined by equation (9). We extend the equation by $\pm(a_m x b_m \phi_m(\theta_m e_x))$, to construct the term α, which is now dependent on e_x, and θ. We are then left with two other terms, namely, $-a_m e$, and $\gamma_m(e_m, e_x)$.

$$\begin{aligned}
\dot{e} =& \dot{x}_m - \dot{x} \\
=& -a_m x_m + b_m\phi_m(\theta_m e_m) - (-ax + b\phi(\theta e_x)) \\
& \pm (a_m x + b_m \phi_m(\theta_m e_x)) \\
=& -a_m \underbrace{(x_m - x)}_{=e} \\
& + \underbrace{(a - a_m)x + b_m\phi_m(\theta_m e_x) - b\phi(\theta e_x)}_{=\alpha(e_x,\theta)} \\
& + \underbrace{b_m(\phi_m(\theta_m e_m) - \phi_m(\theta_m e_x))}_{=\gamma_m(e_m,e_x)}
\end{aligned} \tag{9}$$

$$\begin{aligned}
\dot{V}(e,\alpha) =& 2e(-a_m e + \alpha(e_x,\theta) + \gamma_m(e_m,e_x)) + 2\alpha c\dot{\alpha}(e_x,\theta) \\
=& -2a_m e^2 \\
& + 2e\gamma_m(e_m,e_x) \\
& + 2\alpha(e_x,\theta)(e + c\dot{\alpha}(e_x,\theta))
\end{aligned} \tag{10}$$

By substituting equation (9) into equation (8), we get equation (10).

Note, that the first term $-2a_m e^2$ is always negative, since $a_m > 0$, and $e^2 > 0$. The second term will always be negative, due to the property of $e = e_x - e_m$. It follows that if $e < 0 \Rightarrow \gamma_m(e_m, e_x) > 0$, which implies that the second term is negative, and if $e > 0 \Rightarrow \gamma_m(e_m, e_x) < 0$, which again implies that the second term is negative. Hence, the second term reamins negative.

Since, we can already ensure that the first two terms of equation (10) are negative, it remains to show that the last term is always negative or equal to zero at all times. Since θ is dependent on time, this implies $\dot{\alpha}(e_x, \theta)$ will have a term that includes $\dot{\theta}$. Therefore, we choose $\dot{\theta}$, such that the third term in equation (10) is nullified. Note, that due to the way we construct the update law, stability is guaranteed, since the Lyapunov function is positive definite and simultaneously the decrease condition is satisfied.

$$0 = 2\alpha(e_x,\theta)(e + \dot{\alpha}(e_x,\theta)) \tag{11}$$

Table I: Simulation settings

	Constant	Sinusoidal	Smooth pulse
Learning rate γ	0.05	0.15	0.005
Amplitude	-	1	0.25

$$\begin{aligned}
\dot{\theta} =& \frac{1}{e_x b \frac{\partial\phi(\tilde{x})}{\partial\tilde{x}}|_{\tilde{x}=\theta e_x}} \\
& (\frac{1}{c}e + (a - a_m)\dot{x} + b_m \frac{\partial\phi_m(\tilde{x})}{\partial\tilde{x}}|_{\tilde{x}=\theta_m e_x}\theta_m \dot{e}_x) \\
& - \frac{\theta}{e_x}\dot{e}_x
\end{aligned} \tag{12}$$

When taking a closer look at the update rule in equation (12), it follows that we require to implement an update hold, when $e_x = 0$. This is reasonable, since whenever the state of our reference system is equal to the reference signal, we will not be able to extract information on how to change the weights in order for our system to converge towards the reference signal. In practice this means that we implement a threshold ε, where we do not update the weights.

V. Results

In this section, the simulation results of NMRAC for a first-order system are presented. The example is constructed, to show that the parameters of the NN converge to the desired values, using the proposed algorithm. The NN consists of one neuron, with an nonlinear actiavtion function, as defined in (3). Hence, the goal of the simulation is to show that $\theta \to \theta_m$, while keeping all other parameters constant and equal to one another.

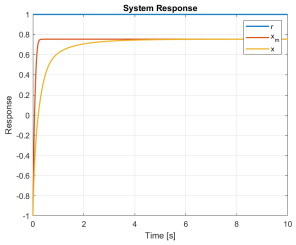
The dynamics are chosen to be $a_m = a = 10$, $b_m = b = 10$, $\theta_m = 4$, and the activation function saturates at a value of 1. The precision threshold ϵ is chosen to be computer precision in all simulations. We compute the weight update as the analytical solution for each time instance. The sampling time for the simulations is $T_s = 0.01$, and the learning rate differs in each simulation to showcase faster and slower convergence. Different simulations for different reference signals are provided, including a constant signal, a sinusodial signal, and a smooth pulse signal. The simluation parameters can be taken from table I.

A. Constant Reference Signal

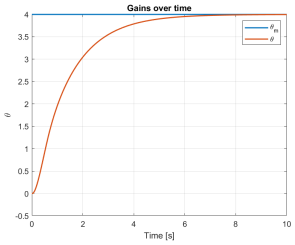
We commence by using a constant reference signal. We choose the system's initial conditions to be at 0. The value of the constant reference signal is 1. This will result in a steady state error for the reference model since we only take a feedback controller into account. However, we will ignore this steady state error, since for us the ability of the controller to learn the behavior of the reference model is of interest. Additionally, we set $\theta_m = 4$ and the initial value of the weight $\theta = 0$. Finally, we simulate the system response for 5 seconds and choose the constant parameter $c = 0.05$.

In Figure 2a, we observe the steady state error, which is at around 0.2. As aforementioned, we will disregard this error,

since we are interested in how well our NNC can learn the behavior of our reference model. To this end, we can observe that the system converges to the reference model within 5 seconds. Furthermore, in Figure 2b, we can see the weights of the NNC and the error e are equal to their desired values, $\theta = 4$ and $e = 0$ respectively. In Figure 3a, we can see that the Lyapunov function converges to zero over time. Additionally, we can see that the derivate of the Lyapunov function is negative and only reaches zero when the Lyapunov function itself converges to zero. This indicates that the proposed learning schematic is stable for the desired reference signal, and we can learn the desired parameter.

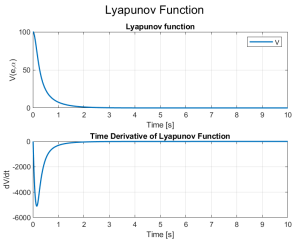


(a) System response

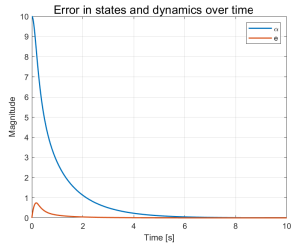


(b) Evolution of gain update

Figure 2: Simulation results of nonlinear MRAC with a constant input



(a) Lyapunov function and its time derivate



(b) Error in internal states and dynamics of the system

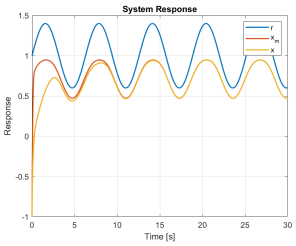
Figure 3: Lyapunov function and convergence of the system w.r.t. internal states and dynamics

B. Sine Signal

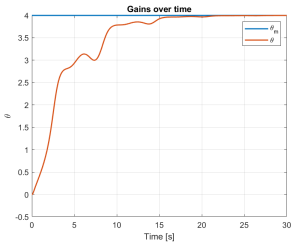
The second reference is a sinusoid signal, with an amplitude of 0.25, to stay in a region that is not affected too much by the saturation function. As before, we choose the initial $\theta = 0$, $\theta_m = 4$, and the initial conditions of both the system and the reference model to be at -1. We choose this specific initial condition only to be able to see the convergence in the plots. Finally, we simulate the system response for 30 seconds and choose the constant parameter $c = 0.15$, which corresponds to a less aggressive learning strategy compared to the constant input signal.

Figure 4a shows the response of the system. We observe that the system state, and reference model state both converge towards the desired reference model within around 25 seconds. Simultaneously, the parameter θ converges to towards the desired value, which can be seen in Figure 4b. Even though the parameter converges, we observe oscillations. This behavior is expected, due to the update law. Since the update law is dependent on $\dot{e}_x$, it is dependent on $\dot{r}$ and $\dot{x}$. We observe that whenever $\dot{e}_x$ switches signs and simultaneously e is small, $\dot{\theta}$ changes sign, and hence the parameter starts oscillating. This behavior can be changed thourgh adapting the learning rate. A more aggressive learning strategy will be showcased in the next example.

In Figure 5a we can see that the Lyapunov function converges to zero over time. Additionally, we can see that the derivate of the Lyapunov function is negative and only reaches zero when the Lyapunov function itself converges to zero. The results of this simulation indicate again that the proposed learning schematic is stable for the desired reference signal, and we can learn the desired parameter.

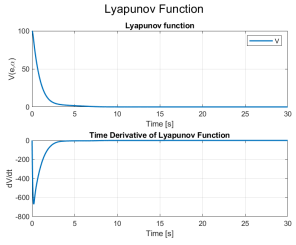


(a) System response

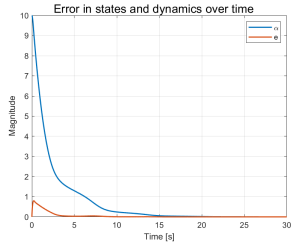


(b) Evolution of gain update

Figure 4: Simulation results of nonlinear MRAC with a sine input



(a) Lyapunov function and its time derivate



(b) Error in internal states and dynamics of the system

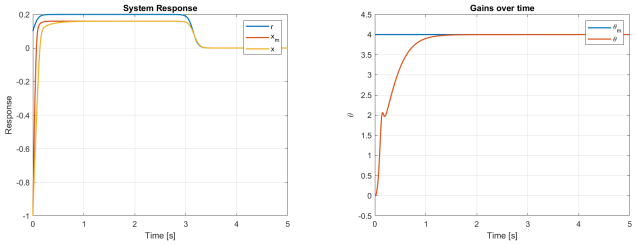
Figure 5: Lyapunov function and convergence of the system w.r.t. internal states and dynamics

C. Smooth Pulse Signal

The third and last input signal we will look at is a pulse signal. More specifically, we will use a smooth pulse signal, because we require the reference signal to be continuous since we require its derivative in the update law. We construct the smooth pulse signal, by using a sine function and applying a smooth saturation function, as defined in equation (3). We choose the signal to have an amplitude of 0.25 and shift the signal such that its minimum value touches upon 0. As before, we choose the initial conditions of the system and plant to be -1. We simulate the system response for 5 seconds and choose the constant parameter $c = 0.005$.

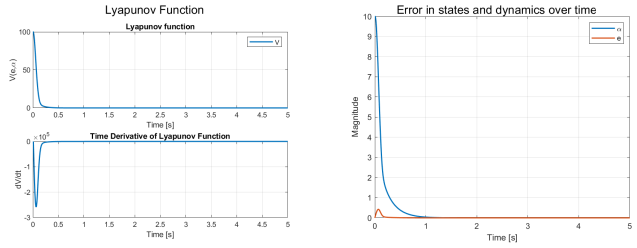
Figure 6a shows the response of the system. We can observe that the system converges towards the desired reference model within just under 1 second. The parameter θ converges to towards the desired value within 2 seconds, which can be seen in Figure 6b. Even though the parameter converges, we observe that it oscillates. This behavior is expected, due to the update law. Since the update law is dependent on $\dot{e}_x$, it is dependent on $\dot{r}$ and $\dot{x}$. We observe that whenever $\dot{e}_x$ switches signs and simultaneously e is small, $\dot{\theta}$ changes sign, and hence the parameter starts oscillating. Similarly, as in the case of the sine input, this behavior can be influenced by decreasing the value of c.

The Lyapunov function and its derivative, as shown in Figure 7a indicate the convergence of the system and the parameters, since V converges to 0 and its derivative is always negative and approach zero when V converges. The results of this simulation indicate that the proposed learning schematic is stable for the desired reference signal, and we are able to learn the desired parameter. The three simulation results strongly indicate that with the proposed update law we are able to learn the desired parameters in the presence of a nonlinear activation function.



(a) System response (b) Evolution of gain update

Figure 6: Simulation results of nonlinear MRAC with a smooth pulse input



(a) Lyapunov function and its time derivate (b) Error in internal states and dynamics of the system

Figure 7: Lyapunov function and convergence of the system w.r.t. internal states and dynamics

VI. Conclusion

The simulation results of the scalar system indicate that the proposed update law successfully enables the system to learn the desired parameters across different reference signals. The speed of convergence can be controlled by adjusting the factor c, with smaller values leading to faster convergence. Additionally, this work identifies numerical instability as a potential issue. This instability can be reduced by lowering the sampling time, thus enhancing the precision of integral and derivative approximations, improving overall simulation accuracy.

Acknowledgment

...

References

[1] H. Whitaker, "An adaptive system for control of the dynamics performances of aircraft and spacecraft", *Inst. Aeronatical Sciences*, pp. 59–100, 1959.

[2] M. Bosshart, G. Ducard, and C. Onder, "Comparison Between Two PID Neural Network Controller Approaches - Simulation and Discussion", in *2021 International Conference on Control, Automation and Diagnosis (ICCAD)*, ISSN: 2767-9896, Nov. 2021, pp. 1–6. DOI: 10.1109/ICCAD52417.2021.9638766. Accessed: Feb. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9638766.

[3] B. Shackcloth and R. L. But Chart, "Synthesis of Model Reference Adaptive Systems by Liapunov's Second Method", *IFAC Proceedings Volumes*, 2nd IFAC Symposium on the Theory of Self-Adaptive Control Systems, Teddington, UK, September 14-17, 1965, vol. 2, no. 2, pp. 145–152, Sep. 1965, ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)69028-1. Accessed: Apr. 24, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017690281.

[4] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks", *Neural Networks*, vol. 3, no. 5, pp. 551–560, Jan. 1990, ISSN: 0893-6080. DOI: 10.1016/0893-6080(90)90005-6. Accessed: May 16, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608090900056.

[5] Y. Jiang, C. Yang, J. Na, G. Li, Y. Li, and J. Zhong, "A Brief Review of Neural Networks Based Learning and Control and Their Applications for Robots", en, *Complexity*, vol. 2017, e1895897, Oct. 2017, Publisher: Hindawi, ISSN: 1076-2787. DOI: 10.1155/2017/1895897. Accessed: Feb. 12, 2024. [Online]. Available: https://www.hindawi.com/journals/complexity/2017/1895897/.

[6] S. Cong and Y. Liang, "PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems", en, *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 3872–3879, Oct. 2009, ISSN: 0278-0046. DOI: 10.1109/TIE.2009.2018433. Accessed: May 23, 2024. [Online]. Available: http://ieeexplore.ieee.org/document/4812095/.

[7] T. D. C. Thanh and K. K. Ahn, "Nonlinear PID control to improve the control performance of 2 axes pneumatic artificial muscle manipulator using neural network", *Mechatronics*, vol. 16, no. 9, pp. 577–587, Nov. 2006, ISSN: 0957-4158. DOI: 10.1016/j.mechatronics.2006.03.011. Accessed: May 23, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957415806000523.

[8] G. Norris, G. J. J. Ducard, and C. Onder, "Neural Networks for Control: A Tutorial and Survey of Stability-Analysis Methods, Properties, and Discussions", en, in *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Mauritius, Mauritius: IEEE, Oct. 2021, pp. 1–6, ISBN: 978-1-66541-262-9. DOI: 10.1109/ICECCME52200.2021.9590912. Accessed: May 6, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9590912/.

[9] H. K. Khalil, *Nonlinear Systems*, en. Prentice Hall, 1996, Google-Books-ID: qiBuQgAACAAJ, ISBN: 978-0-13-228024-2.