**Doctoral Dissertation**

# Synthesis of Practical Noise-shaping Quantizers for Networked Control Systems

Rodriguez Ramirez Juan Esteban

March 15, 2019

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Rodriguez Ramirez Juan Esteban

Thesis Committee:

| | |
|---|---|
| Professor Kenji Sugimoto | (Supervisor) |
| Professor Shoji Kasahara | (Co-supervisor) |
| Associate Professor Takamitsu Matsubara | (Co-supervisor) |
| Associate Professor Yuki Minami | (Co-supervisor, Osaka University) |
| Assistant Professor Masaki Ogura | (Co-supervisor) |

# Synthesis of Practical Noise-shaping Quantizers for Networked Control Systems*

Rodriguez Ramirez Juan Esteban

**Abstract**

Nowadays networked control systems (NCSs) are being widely implemented in many applications. There are several problems that negatively affect and compromise the design of practical NCSs. Some of these problems are (i) data rate constraints of the channel, (ii) network traffic congestion, and (iii) inaccuracies in the model of the plant. The aim of this thesis is to develop novel noise-shaping quantizers for NCSs and their design methods that alleviate the effects of these problems. These quantizers filter the quantization noise and convert continuous-valued signals into the appropriate discrete-valued ones. First, an improved metaheuristic based design is proposed for finite-level dynamic quantizers that minimize the performance degradation caused by quantization in systems subjected to data-rate constraints. Second, in order to deal with the network traffic congestion, a switching type dynamic quantizer is proposed. This quantizer is actuated by a Gaussian functions based event-generator attached to the plant and sends the data only when needed. Third, for the situations in which the model of the plant is absent or is unreliable, this study proposes a type of quantizer implemented with neural networks and a time series of the plant's inputs and outputs. This quantizer does not need a model of the plant, and could be applied to time invariant linear or nonlinear systems. The designs of these quantizers are formulated as nonlinear and nonconvex optimization problems that cannot be solved using conventional optimization techniques. Therefore, this thesis proposes design methods based on covariance matrix adaptation evolution strategy

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, March 15, 2019.

and differential evolution, which are stochastic optimization algorithms. The effectiveness of these quantizers and their design methods are verified by means of a plethora of numerical examples. In addition, their performances are compared among each other using statistical analysis tools. Several conclusions are reached from these simulations and several solutions are developed to improve the performance of these quantizers.

**Keywords:**

Networked Control System (NCS), Dynamic Quantization, Communication Constraints, Network Traffic Reduction, Metaheuristics, Neural Networks

# Contents

**1. Introduction** **1**
1.1 Networked Control Systems . . . . . 1
1.2 Quantizer Design for Networked Control Systems . . . . . 4
1.3 Noise-shaping Quantizers . . . . . 5
1.4 Motivation and Objective . . . . . 8
1.5 Contributions and Philosophy . . . . . 10
1.6 Dissertation Overview . . . . . 12
1.7 Notations . . . . . 13

**2. Noise-shaping Quantizers Design Problems** **14**
2.1 Performance Evaluation of Noise-shaping Quantizers . . . . . 14
2.1.1 Performance Index . . . . . 16
2.1.2 Data Rate Constraints Management . . . . . 17
2.2 Design Problem and Constraints . . . . . 18
2.3 Metaheuristics . . . . . 20
2.3.1 Covariance Matrix Adaptation Evolutionary Strategy . . . 21
2.3.2 Differential Evolution . . . . . 24
2.3.3 Firefly Algorithm . . . . . 25

**3. Design of Finite-level Quantizers Under Data Rate Constraints** **27**
3.1 Considered System . . . . . 27
3.2 Performance Index . . . . . 28
3.3 Minimum Quantization Interval . . . . . 30
3.4 Design Problem . . . . . 34
3.5 Design Variable Setting . . . . . 35
3.6 Numerical Examples . . . . . 36
3.7 Comparison Among Metaheuristics . . . . . 41
3.7.1 Success Rate Comparison . . . . . 41
3.7.2 Convergence Time Comparison . . . . . 44
3.8 Summary . . . . . 45

**4. Design of Event-triggered Quantizers (ETQs) 48**
4.1 Introduction . . . 48
4.2 Problem Formulation . . . 50
4.2.1 System Description . . . 50
4.2.2 Performance Index . . . 52
4.2.3 Data Rate Constraint . . . 54
4.2.4 Network Utilization Rate . . . 54
4.2.5 Quantizer Design Problem . . . 55
4.2.6 Event Definition . . . 56
4.3 Numerical Examples . . . 57
4.4 Comparison among different versions of ETQ . . . 60
4.5 ETQ for Multiple Input Signals . . . 65
4.6 Summary . . . 69

**5. Design of Neural Network Quantizers (NNQs) 71**
5.1 Introduction . . . 71
5.2 Feedforward Neural Networks . . . 73
5.2.1 Structure . . . 73
5.2.2 Initialization . . . 76
5.3 Regression Based NNQ . . . 77
5.3.1 System Description . . . 77
5.3.2 Training Data . . . 79
5.3.3 Performance Index . . . 80
5.3.4 Design Problem . . . 81
5.4 Classification Based NNQ . . . 82
5.5 Numerical Examples . . . 84
5.5.1 Regression and Classification Based NNQ Comparison . . . 87
5.5.2 Comparison with Linear Dynamic Quantizers . . . 91
5.5.3 Activation Functions Comparison . . . 93
5.6 NNQ for Multiple Input Signals . . . 96
5.6.1 Noise Addition Effect . . . 98
5.6.2 Performance Indexes Comparison . . . 102
5.7 Summary . . . 105

**6. Conclusion** **107**
6.1 Summary . . . . . 107
6.2 Limitations . . . . . 109
6.3 Possible Research Directions . . . . . 109

**Acknowledgements** **111**

**References** **112**

**Appendix** **127**

**A. Default Parameters for the $(\mu/\mu_W, \lambda)$ CMA-ES Algorithm** **127**

**B. Particle Swarm Optimization** **128**

**C. NNQs Statistical Analysis Details** **129**

# List of Figures


1 General structure of an NCS. . . . . 1
2 Example of a finite-level quantizer . . . . . 3
3 Structure of an open loop quantized control system. . . . . 4
4 Feedback type dynamic quantizer . . . . . 6
5 Thesis philososhy schematic . . . . . 11
6 Error system . . . . . 14
7 Static quantizer's example . . . . . 16
8 Penalty cost comparison . . . . . 20
9 CMA-ES operation's example over a two dimensional problem . . 22
10 Designed quantizers' operation example . . . . . 43
11 Success rate for the second order plant ($P_1$). . . . . 44
12 Success rate for the third order plant ($P_2$). . . . . 44
13 Metaheuristics' convergence behavior for the second order plant $P_1$. 46
14 Metaheuristics' convergence behavior for the third order plant $P_2$. 47
15 Event-triggered quantizer scheme . . . . . 50
16 Operation of the trigger signal $\sigma(k)$ . . . . . 51
17 Trigger function $H(y_q)$ example . . . . . 52
18 ETQ's error system . . . . . 53
19 ETQ's example resulting signals . . . . . 58
20 GETQ resulting error signals . . . . . 59
21 ETQ's error signal comparison for different $\Psi^*$s . . . . . 60
22 Alternative operation of the ETQs . . . . . 61
23 ETQ alternative design method's output signals $y(k)$ . . . . . 63
24 Input and trigger signals for ETQ designed with multiple $u(k)$s . 67
25 Outputs signals for ETQ designed with multiple $u(k)$s . . . . . 68
26 3 layered feed-forward neural network example . . . . . 73
27 Hidden layers' activation function $\mathsf{h}(\cdot)$ comparison . . . . . 75
28 Neural network quantizer's considered system . . . . . 77
29 Static quantizer's example . . . . . 78
30 Quantizer's supervised learning . . . . . 79
31 NNQ's error system . . . . . 80
32 NNQ design approach differences . . . . . 82

33 Static quantizer's variation example . . . . . 83
34 Resulting signals from designed NNQs for $M = 2$ and $n_L = 2$ . . . 86
35 Output signals from designed NNQs for $M = 2$ and $n_L = 4$ . . . . 87
36 Output signals from designed NNQs for $M = 8$ . . . . . 87
37 $Q_{NNR}$ and $Q_{NNC}$ cumulative errors comparison . . . . . 91
38 $Q$ and $Q_{NN}$ cumulative errors comparison . . . . . 92
39 3-way ANOVA main effects plot for the hs comparison . . . . . 96
40 3-way ANOVA interaction plot for the hs comparison . . . . . 97
41 Error system signals with a NNQ trained for multiple $u(k)$s . . . . 99
42 Error system signals with a NNQ trained for a single $u(k)$ but fed with multiple $u(k)$s . . . . . 100
43 Output signals comparison for NNQs trained for multiple $u(k)$s and different performance indexes . . . . . 104

# List of Tables


1 Hyperparameters used in the simulations . . . . . 38
2 Simulation results for the second and third order plants by CMA-ES 39
3 Simulation results for the second and third order plants by DE . . 40
4 Simulation results for the second and third order plants by FA . . 41
5 Simulation results for the second and third order plants by PSO . 42
6 Considered ETQ's design methods . . . . . 61
7 Minimum $\tilde{E}$ for the different ETQ's design methods . . . . . 65
8 Considered neural networks' structure. . . . . 85
9 $E(Q_{NN})$ analysis for $\mathsf{h} = \text{sigmoid}$ . . . . . 88
10 Tukey pairwise comparison for $\mathsf{h} = \text{sigmoid}$ and single factors . . . 90
11 $Q$ and $Q_{NN}$ cumulative errors comparison . . . . . 93
12 $E(Q_{NN})$ results summary for $M = 8$, $\mathsf{h} = \tanh$ and $\mathsf{h} = \text{ReLU}$ . . 94
13 3-way ANOVA Tukey test results for $\mathsf{h}$ comparison . . . . . 95
14 Noise effects analysis for NNQs designed for multiple $u(k)$ . . . . 101
15 Performance indexes comparison for NNQs designed for multiple $u(k)$ . . . . . 103
16 Tukey pairwise comparison 1-way ANOVA for performance indexes comparison . . . . . 105
17 $E(Q_{NN})$ 3-way ANOVA for $\mathsf{h} = \text{sigm}$ . . . . . 129
18 Tukey pairwise comparison 3-way ANOVA for $M = 2$ . . . . . 130
19 Tukey pairwise comparison 3-way ANOVA for $M = 8$ . . . . . 131
20 3-way ANOVA for $\mathsf{h}$ comparison . . . . . 132
21 3-way ANOVA Tukey test results for $\mathsf{h}$ comparison . . . . . 133

# 1. Introduction

## 1.1 Networked Control Systems

In recent years, the networked control systems or NCSs have been receiving a lot of attention from researchers and manufactures due to their many advantages and potential applications [46, 72]. In an NCS the plants, controllers, sensors and actuators are physically separated and connected to each other via communication networks as it is shown in Figure 1. A characteristic feature of the NCSs is that the communication network is shared and multiple control systems can be operating over it at the same time. Moreover, the NCS can be connected to the companies intranet or even to the internet allowing any type of data to go through them, not only control signals [3, 133, 143].

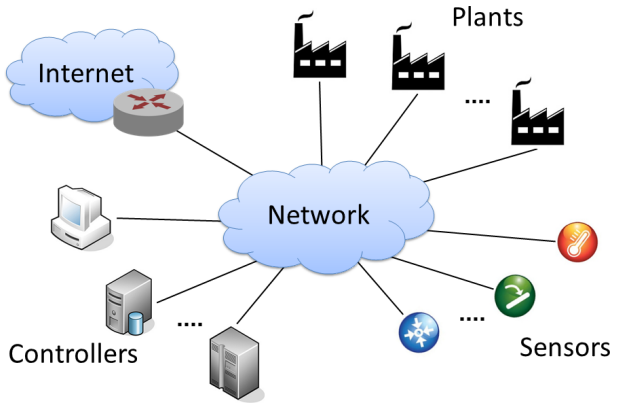



Figure 1: General structure of an NCS.

The NCSs have several advantages. The elimination of the unnecessary wiring and the use of network technologies reduce the system's complexity and make it very easy to add or withdraw controllers, sensors or actuators. Thus, NCSs make the systems very flexible and scalable. Additionally, by using a network to connect the elements of the control system, the costs of installation and maintenance are greatly reduced. The NCSs connect the cyber space to the real world making the tele-operation of systems very easy to implement [31, 145]. Examples of NCSs are found in industrial automation [59], control systems for automobiles and aircraft [118, 119], mobile sensor networks [88], haptics collaboration over the Internet [45, 47, 121] and the control of large distributed systems such as

smart grids [114, 57], transportation and water distribution networks. They have a lot of potential applications like disaster prevention systems [14], hazardous environments monitoring, domotics [124], remote surgery [75], distribured robotic networks [12], space and terrestrial exploration, tele-operated systems [16] and many others [81, 36].

Although NCSs bring many advantages to the systems, they have several problems associated to them, like (i) data-rate constraints in the channels, (ii) network traffic congestion, quantization errors due to (i), and packet drops due to (ii) among others [144, 35]. In addition, the (iii) inaccuracies in the model of the plant make necessary to build more robust NCSs capable of dealing with noise and uncertainties [42]. This thesis focuses in ways to alleviate the effects of these problems.

In NCSs, the control and sensor signals are transmitted through the communication network. These signals need to be quantized in order to be sent through a digital channel. The quantization is the process in which the continuous-valued signals are transformed into discrete-valued ones. The device that performs the quantization of the signals is called quantizer. In general, a quantizer is composed by a certain *number of quantization levels* $M$ and the *quantization interval* $d$, which is the distance between the quantization levels. In many studies the value of $M$ is unbounded ($M \to \infty$) [53, 11, 76, 56, 93]. In others, the quantizers that have a finite number of quantization level, these quantizers are called finite-level quantizer [90, 116, 140]. In regard to $d$, the quantization interval can be a constant or it can be variable, following some type of rule, like in the logarithmic quantizers [41] and in the hysteresis quantizers [146]. Figure 2 shows an example of a finite-level quantizer with constant $d$. When the effects of quantization are considered in the design and operation of a control system, we are in the presence of a quantized control system. Figure 3 shows how an open loop quantized system works.

There exist limitations in the amount of data that can be transmitted per unit of time in the networked control systems. These limitations are known as *data rate constraints* of the channel. Since the quantizer is supposed to send data to the plant in each sampling time, the data rate constraint of the channel limits the number of quantization levels in the quantizer [127, 86]. Each quantized signal is

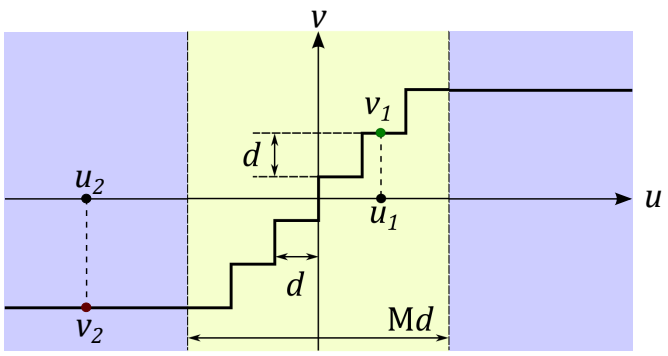



Figure 2: Example of a finite-level quantizer ($M = 6$). Where $u$ is the continuous-valued input and $v$ is the discrete-valued output. The yellow area represents the *range of the quantizer* and the blue one is the *saturation area*. In order for the quantizer to work properly, $u(t)$ should be always inside the range of quantizer. For example the value of $u_1$ gives a quantized value $v_1$ inside the quantizer's range, but the value $u_2$ inside the saturation area gives a quite smaller quantized value $v_2$.

transformed into a chain of bits by an encoder in order to be transmitted. The amount of bits generated by the encoder depends on the amount of quantization levels that the quantizer has. These bits should be sent before another sampled signal arrives to the encoder. Then, the system will work properly only if the amount of bits that can be sent through the channel during a sampling time interval is more or at least equal to the amount of bits generated by the encoder due to one quantized signal [90, 116].

Furthermore, a finite number of quantization levels may cause the saturation of the amplitude of the quantized signal. The saturation happens when the input signal's amplitude is bigger than the maximum value that the quantizer can transform. What happens in this case is usually that the quantizer assigns to the quantized output the maximum value that the quantizer can produce. In consequence there may be a big difference between the input and the output of the quantizer. Such saturation problem has the potential to destabilize the systems [131, 132, 34].

Since nowadays there are networks that transmit data in gigabits per second,

the data rate constraints may not look like a big problem. But remember that the NCSs are shared networks and the effects of congestion and bottle necks will affect the performance of the system [86].

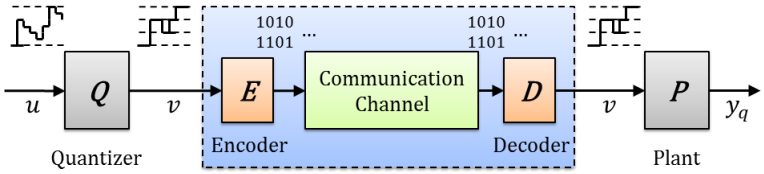



Figure 3: Structure of an open loop quantized control system. The quantizer $Q$ transforms the control signal $u$ in the quantized signal $v$. Then, $v$ is transformed into a chain of bits by the encoder $E$ in order to be sent through the channel and it is reconstructed on the destination by a decoder $D$.

## 1.2 Quantizer Design for Networked Control Systems

There are several studies that tackle the problem of networked control systems subjected to quantization. These studies consider different types of quantizers that are designed for different type of systems. The quantizers can be classified into static or dynamic. In the static quantizers the quantized output depends only on the current input. They are memoryless. On the other hand, the dynamic quantizers produce a quantized output that is function of the current input and past outputs. They have an internal memory. Many of these quantizers are designed along side with the controller in order to secure the system's stability. Examples of these quantizers are the following.

Static quantizers designed to achieve stabilization considering the minimium data rates were developed for noiseless linear system [132, 25, 22] and using robust quantization [64, 63]. In other studies the designed of the coarsest static quantizers for stabilization [24, 68, 55] and for identification [129, 54] are considered. In addition, there is the sector bound approach in the design of static logarithmic quantizers were they are designed to facilite the quadratic stability of the system [27, 29] and for finite gain stabilization [142].

On the side of the dynamic quantizers there are also many studies. The ones that consider the minimum data rates for stabilization using deterministic setups

were developed for linear systems [82, 83, 97, 115] and for nonlinear systems [85, 20]. Furthermore, there are the ones that were developed in stochastic setups [126, 128, 84]. Then, there are the studies that consider the coarsest dynamic quantizer for stabilization. For instance, the quantizers with zoom-in and zoom-out strategy. These quantizers dynamically adjust the quantizer's range so that it increases or decreases as the plant state approaches or diverges from the target, respectively. These quantizers have been developed for linear systems [11, 127, 66, 67] and for nonlinear ones. [20, 65]. Also, in this category can be found the finite-level logarithmic quantizer [28].

Another category of quantizers are the ones designed for event-driven control. In this category the receding horizon quantizers can be mentioned. These dynamic quantizers are designed by solving a quantized finite horizon optimization problem [101, 102, 103]. The solution can be implemented by means of a vector quantizer or by a polytopal partition of the state space. Also, in this category the delta modulators can be found [21, 56]. These quantizers use a differential coding scheme to minimize the number of bits to be transmitted through the channel.

Finally, there are the feedback type dynamic quantizers that are designed in the context of networked control to minimize the system's performance degradation [4, 5, 90, 116, 140]. This dissertation focuses partially in this type of quantizers. Some characteristics of these quantizers that distinguished them from the ones mentioned above include that the quantization interval $d$ is constant, they are built using a digital filter followed by a static quantizer, and that they are designed independently from the controller. Following, these quantizers will be explained in more detail.

## 1.3 Noise-shaping Quantizers

The difference between a signal and its quantized version is known as quantization error. The quantization errors are reflected in the output of the plant causing the degradation in the performance of the system. The performance degradation due to quantization might lead to instability. Several studies have been carried out aiming to reduce the performance degradation due to quantization, and they have shown that properly designed feedback type dynamic quantizers are effective for this task [4, 5, 117, 77, 141]. This feedback type of quantizers can be grouped

into a bigger category of quantizers known as noise-shaping quantizes.

This dissertation considers the design of practical feedback type noise-shaping quantizers $Q$. These quantizers are composed of a digital filter followed by a static quantizers q as it is shown in Figure 4. The noise-shaping quantizers filter the continuous-valued input signals $u(k)$ along side with past discrete-valued outputs $v(k)$ to produce shaped noise signals $\bar{u}(k)$. These signals are fed to the static quantizer that maps them onto discrete-valued outputs $v(k+1)$.

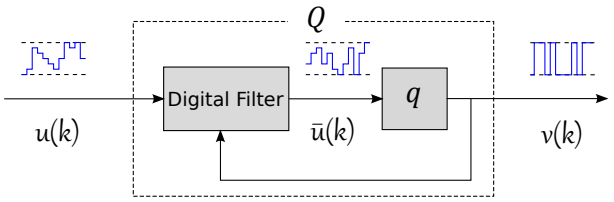



Figure 4: Feedback type noise-shaping quantizer.

These quantizers are called noise-shaping quantizer, because the filter *shapes* the input and output signals together producing a noise like signal that is applied to the static quantizer. As the previously discussed quantizers, the noise-shaping quantizers can be static or dynamic. They are considered dynamic when the digital filter have a memory that stores the quantizer's internal state. This state changes over time and affects the outputs. The quantizers are considered static when the filter does not have such internal memory.

The limited amount of quantization levels in the quantizer may cause the quantization errors to increase. For example, when the range of the quantizer is big in comparison with the range of the input signal, the number of quantization levels is small. The quantization errors and, in consequence, the performance degradation can be reduced greatly by the proper design of the quantizers. In the noise-shaping quantizer design the goal is usually to make the quantization errors as small as possible in order to approximate the quantized system to the ideal system without quantization.

In past studies some authors focuse only in the design of the quantizer's digital filter assuming that the quantization interval $d$ is given and that the quantizer has an infinite number of quantization levels $M$. For example, Azuma and Sugie

found a closed form expression with respect to the plant parameters to evaluate the performance of a class of dynamic quantizers for linear and for non-linear time invariant systems[4, 6]. Besides, based on the performance analysis, they provided an analytic expression of the optimal quantizer and its performance. Minami and Muromaki used differential evolution (DE) for the design of fixed-order decentralized dynamic quantizers for discrete-valued input control [78].

In those studies the effects of the channel's data rate constraints were not taken into account. Then, in order to consider these effects it is necessary to design the digital filter and the static quantizer at the same time. In this case the static quantizer has a finite number of quantization levels. Some studies have tackled already this type of problem [90, 116, 140, 106] and developed methods for the design of the finite-level noise-shaping quantizers. However, these methods are not perfect because they may not give optimal solutions to the quantizer design problems.

For instance, Okajima et al. proposed a design method for finite-level dynamic quantizers for SISO systems [90]. They derived a design method of the quantizer's smallest quantization interval. Based on the invariant set analysis the design method is derived as a linear matrix inequality (LMI) problem. In [116], Sawada et al. extended the previous result to consider MIMO systems. In these two studies the authors solved a relaxed version of the original quantizer design problem and because of that the solution might be conservative. Then, Yoshino et al. presented a design method for a dynamic quantizer under data rate constraints [140]. The design is carried out by optimizing an objective function using a metaheuristic known as particle swarm optimization (PSO). This method gives a smaller performance degradation than the method proposed by Okajima et al., but the rate of success in finding the optimum values of the quantizer is very low. The low success rate is explained by the existence of local minima in the objective function and the tendency of PSO of getting trapped on it.

Despite all these development in regard to quantization there are still many drawbacks and open problems that need be addressed. One of these problems is the conservativeness of the proposed solutions for the design of dynamic quantizer for NCSs affected with data rate constraints [90, 116]. In addition, none of the previous studies have considered the effect that the network traffic congestion

has on the system performance or the possibility that the network traffic can be regulated by noise-shaping quantizers. Moreover, another drawback of the quantizers proposed previously is that most, possibly all, of them are model-based designs. This means that their design is carried out using a model of the considered plant. There are cases, however, in which the model of the plant is not available or the given model is not reliable. For these cases it is necessary to have a model-free design method for noise-shaping quantizers.

## 1.4 Motivation and Objective

NCSs are a natural step in the evolution of the control systems. They have a lot of advantages and the potential to become the standard in automation. For instance, they are easy to implement and cheap, thus, very likely to be implemented even in small industries. Thereby, it is important to increase the reliability of NCSs reducing the performance degradation caused by quantization. A way to do this is the use of noise-shaping quantizers properly designed to minimize the system's performance degradation.

In addition, the constraints in the communication channel is an issue that needs to be addressed. Even if the channel has a big bandwidth the effects of congestion and bottleneck cannot be ignored, since they make the system unable to operate properly. Then, the quantizers should be designed considering the constraints in the communication channel and in order to put as small traffic as possible in the network.

Moreover, the design of optimal noise-shaping quantizers for NCSs should be carried out even when the model of the plant is inaccurate or totally absent. In other words, a model-free design for optimal noise-shaping quantizers is requited.

For these reasons the main objective of this thesis is:

> To develop practical finite-level noise-shaping quantizers with their respective easy-to-use design methods that:
>
> i Minimize the system's performance degradation due to quantization,
>
> ii Satisfy the channel's data rate constraints,
>
> iii Reduce the traffic that the system puts in the network, and

iv Can be designed without the need of a model of the plant.

In particular, this thesis considers the design of three different types of quantizers that addresses the requirements mentioned above. These quantizers are the following. First, novel metaheuristic based design methods are proposed for finite-level dynamic quantizers that minimize the system's performance degradation due to quantization and satisfy the channel's data rate constraints. Second, a switching type finite-level dynamic quantizer is proposed. This quantizer, known as *event-triggered quantizer* (ETQ), minimizes the system's performance degradation caused by quantization, satisfies data rate constraints, and reduces the traffic that the system puts in the network. This quantizer operates in an event-based manner, it is designed to send data to the plant only when it is needed. The use of this type of quantizers is oriented to systems in which network resources like bandwidth and energy are limited. This quantizer makes a trade-off between the performance of the system and the network traffic generated. Finally, a neural network based finite-level noise-shaping quantizer is developed. This quantizer, called *neural network quantizer* (NNQ), minimizes the performance deterioration of the system, and is designed without using the model of the plant. Instead, time series of the plant's outputs with their respective inputs are employed in the design. This quantizer is useful in the cases in which a model of the plant is not available or it is unreliable. Also, it is used when the model of the plant is too complex to implement an optimal quantizer based on such model. Each of these quantizers are developed in a dedicated chapter of this thesis.

The design problems of the quantizers considered in this thesis are not easy to solve. They lead to the optimization of nonlinear and nonconvex functions, that cannot be solved by algebraic methods or conventional numerical optimization methods such as linear programming or quadratic programming. Thus, the use of metaheuristics seems a reasonable option. Three different metaheuristics were selected to implement the quantizers design methods, namely, covariance matrix adaptation evolution strategy (CMA-ES), differential evolution (DE) and firefly algorithm (FA). The effectiveness of the proposed quantizers and the design methods developed are verified by means of numerical examples. The efficiency of the design methods are compared among each other and with previously developed design methods based on particle swarm optimization (PSO) [140]. At last, com-

parisons among the performance of several variations of the proposed quantizers were carried out and using statistical analysis techniques several conclusion were reached.

## 1.5 Contributions and Philosophy

The main contributions and achievements of this work are summarized as follows:

- The development of practical design methods based on CMA-ES and DE for dynamic quantizers affected by data rate constraints in the channel.
- The comparison of the proposed design methods with a previously developed design method and the evidence of their superior performance.
- The development of a novel event-triggered dynamic quantizer (ETQ) designed for network traffic reduction.
- The introduction of a new parameter, known as network utilization rate, to evaluate the traffic in the network.
- The development of design methods for ETQs based on DE.
- The comparison of the performance among variations of the ETQs and their design methods.
- The introduction of the concept of neural network based noise-shaping quantizer (NNQ).
- The development of a model-free design method for NNQs based on DE.
- The introduction of several variations of NNQ and the comparison among them.
- The verification of the proper operation of the proposed quantizers and their design methods.

Previous studies, like the ones introduced in Section 1.2, present results on quantizer design for NCSs that are not sufficient for applying them to real-world

problems because they mainly focused on a *theoretical approach*. Motivated by this fact, this study aims to develop *practical* noise-shaping quantizers. In this case by *practical* it is understood that the quantizers are developed aiming toward real-world applications.

Following this idea, this dissertation starts by considering previous studies that are theoretical in nature, like the works of Azuma and Sugie, [4, 5] and Okajima et al. [90, 92], and built on top of them expanding the quantizers functionality in two directions or *axes*. These axes are: (1) the reduction of the systems data-rate and (2) the complexity of the target systems. This idea is illustrated in Figure 5.

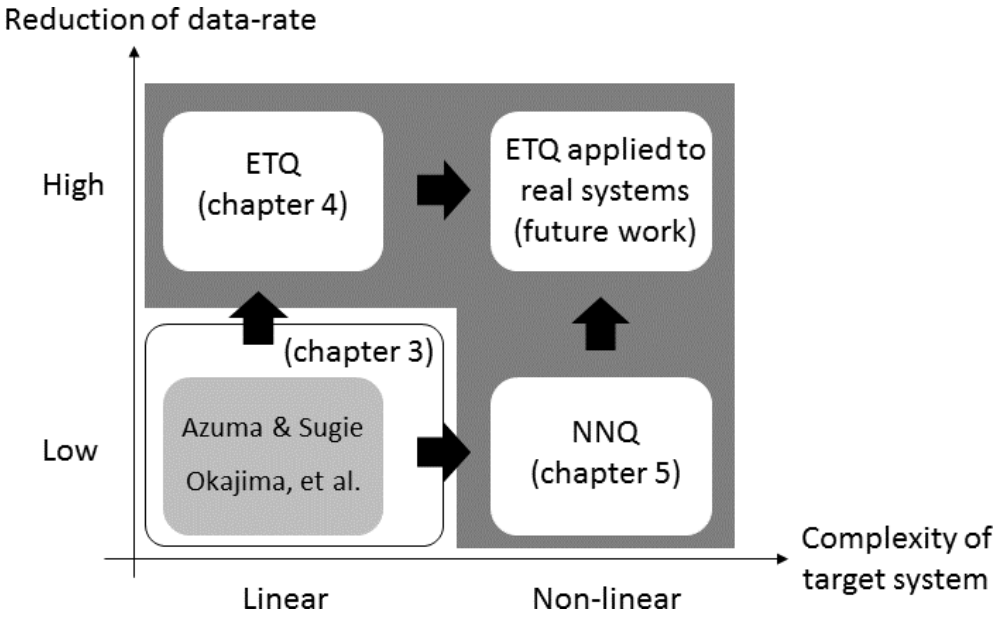



Figure 5: Schematic of the philosophy of this thesis.

As it is shown in Figure 5, this study develops the ETQ, the NNQ, and their design methods based on stochastic optimization. Each of these quantizers expands the capabilities of previous studies in one of the considered axes. With many simulations the practicality of the proposed quantizers was verified. After that, based on the developments made in these two axes the combination of them can produce quantizers that could be applied to real-world problems, which are located in the area between the ETQs and NNQs developments.

The key contributions of this study to the realm of NCSs are (i) the introduction of novel structures of quantizers and (ii) their respective stochastic optimization based design methods.

Furthermore, the contributions of this study make the quantizer design methods more practical; namely, the results help to apply quantizer design methods to real-world problems. For instance, in large scale NCSs composed of a large number of plants the ETQs could be implemented, or in plants with nonlinearities which cannot be modeled the NNQs will be useful.

## 1.6 Dissertation Overview

The rest of this dissertation is organized as follows:

**Chapter 2** presents a digest of the considered noise-shaping quantizers design problems and proposed solutions. Also, the metaheuristics employed to implement the quantizer design methods are described.

**Chapter 3** discusses the finite-level dynamic quantizer affected by data rate constraints. Its metaheuristic based design method is formulated alongside with the methodology and settings to implement it. After that, the effectiveness of the proposed design methods are verified by means of numerical examples and a comparison is carried out among them and previously developed methods.

**Chapter 4** introduces the event-triggered quantizer. Its structure is described, the design problem is formulated and a design method is developed. Then, numerical examples are carried out and the results commented. At last, comparisons among this quantizer variations and their design methods are performed.

**Chapter 5** introduces the concept of neural network quantizers. First, the structure of feedforward neural networks are presented. Then, the NNQs structure is described, and its design method is developed. After that, its effectiveness is verified with numerical simulations and comparisons are carried out among variations of this quantizer.

**Chapter 6** presents final conclusions, the limitations of the study and some perspectives for future expansions of the current work.

## 1.7 Notations

Let $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}_+$ the positive real numbers and $\mathbb{N}$ the natural numbers. For the matrix $\mathbf{A} := [A_{ij}]$, let the matrix abs($\mathbf{A}$) be defined by abs($\mathbf{A}$) := $[|A_{ij}|]$. For a square matrix $\mathbf{A}$, let $\Lambda_i(\mathbf{A})$ denote the $i^{th}$ eigenvalue of $\mathbf{A}$. For a real vector $\boldsymbol{v} := [v_i]$ the expression $\|\boldsymbol{v}\|$ represents the Euclidean norm of $\boldsymbol{v}$. In addition, $\mathbf{I}$ denotes the identity matrix, $\mathbf{0}$ denotes the null matrix, and $\mathsf{E}\|\mathrm{P}\|$ denotes the expected value of a probability distribution P.

# 2. Noise-shaping Quantizers Design Problems

This chapter introduces the design problems of the noise-shaping quantizers that are considered in this dissertation. These quantizers are:

i Finite-level dynamic quantizers affected by data rate constraints,

ii Event-triggered quantizer, and

iii Neural network quantizer.

In addition, the basics on the analysis of these quantizers are given and the meta-heuristics used in their design are described. These quantizers are all developed to a great extent in their respective chapters. Here a digest of these quantizers alongside with their commonalities are exposed.

## 2.1 Performance Evaluation of Noise-shaping Quantizers

In this thesis, the performance of finite-level noise-shaping quantizers is evaluated using a type of system called *error system*. The general form of this error system is shown in Figure 6.

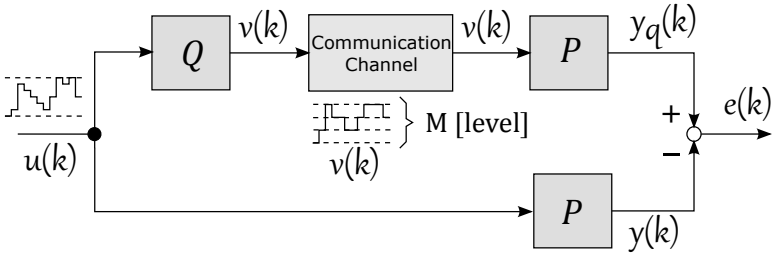



Figure 6: Error system.

This error system is composed of the noise-shaping dynamic quantizer $Q$, the plant $P$ and the channel. The input signal $u(k)$ is applied to the system and it goes through two branches. The lower branch represents the ideal case in which $u(k)$ is applied directly to the plant giving the ideal output $y(k)$. In the upper branch the effects of the channel and quantization are considered, the quantized input signal $v(k)$ is applied to the plant and the output $y_q(k)$ will be known as

real output. The difference between $y(k)$ and $y_q(k)$ is the error signal $e(k)$, which gives a measurement of the performance degradation of the system.

The error system varies slightly from each considered noise-shaping quantizer in this thesis, but the main idea of comparing the ideal case with the case with quantization remains the same.

The assumptions made in this thesis for all the quantizers are the following: the communication channel has no losses and no delays, the plant $P$ is stable and the input signal is bounded, i.e., $u(k) \in U$ for $U := [u_{min}, u_{max}]$.

The noise-shaping quantizers in this thesis are designed for different type of plants. The finite-level dynamic quantizer under data rate constraints and the event-triggered quantizer are designed for linear plants meanwhile the neural network quantizer is designed for any type of plant, linear or nonlinear. Nevertheless, what these plants $P$ have in common is that they are discrete-time, single-input-single-output (SISO), and stable. The structure of the considered linear plants is represented as follows

$$P : \begin{cases} \boldsymbol{x}(k+1) & = \mathbf{A}\boldsymbol{x}(k) + \mathbf{B}u(k), \\ y(k) & = \mathbf{C}\boldsymbol{x}(k), \end{cases} \tag{1}$$

where $k \in \{0\} \cup \mathbb{N}$ is the discrete time, $\boldsymbol{x} \in \mathbb{R}^{n_P}$ is the state vector, $u \in \mathbb{R}$ is the control signal, $y \in \mathbb{R}$ is the output signal, $\mathbf{A} \in \mathbb{R}^{n_P \times n_P}$, $\mathbf{B} \in \mathbb{R}^{n_P \times 1}$ and $\mathbf{C} \in \mathbb{R}^{1 \times n_P}$ are constant matrices and the initial state of the plant is $\boldsymbol{x}(0) = \boldsymbol{x}_0$ for $\boldsymbol{x}_0 \in \mathbb{R}^{n_P}$.

The structures of the considered noise-shaping quantizers are different from each other. However, they all essentially have a structure similar to the one shown in Figure 4. The digital filter is implemented differently in each case but the static quantizer is the same. The structure of the original finite-level dynamic quantizer is given as follows

$$Q : \begin{cases} \boldsymbol{\xi}(k+1) & = \boldsymbol{\mathcal{A}}\boldsymbol{\xi}(k) + \boldsymbol{\mathcal{B}}(v(k) - u(k)), \\ v(k) & = \mathrm{q}[\boldsymbol{\mathcal{C}}\boldsymbol{\xi}(k) + u(k)], \end{cases} \tag{2}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{n_Q}$ is the state vector, $v \in \{\pm\frac{d}{2}, \pm 2\frac{d}{2}, \cdots, \pm\frac{M}{2}\frac{d}{2}\}$ is the quantized output, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_Q \times n_Q}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_Q \times 1}$ and $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{1 \times n_Q}$ are constant matrices, and the initial state of the quantizer is $\boldsymbol{\xi}(0) = \mathbf{0}$. The static quantizer $\mathrm{q}[\cdot]$ rounds

the signals to the nearest quantization level at each time. The parameters of the static quantizer are the number of quantization levels $M \in \mathbb{N}$ and the quantization interval $d \in \mathbb{R}_+$. Figure 7 shows an example of the function q[·] where $M = 6$. Given the structure of considered static quantizer $M$ is restricted to be even.

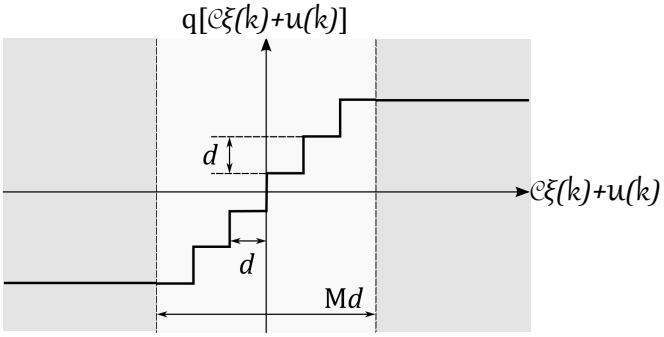



Figure 7: Example of static quantizer q[·] ($M = 6$).

This structure of the dynamic quantizer is also used for the event-triggered quantizer. In the neural network quantizer the digital filter is implemented using a neural network. These quantizers will be explained in detail in their respective chapters.

### 2.1.1 Performance Index

As it was mentioned previously, the error signal $e(k) = y_q(k) - y(k)$ is used to evaluate the performance degradation of the system. By minimizing $e(k)$, the system composed of the quantizer $Q$ and the plant $P$ can be optimally approximated to the plant $P$, in terms of the input-output relation.

In this context a parameter known as *performance index* is used to measure the performance degradation of the system. This index can be defined in many ways, and different types of quantizers and systems may find some definitions more suitable than others. For instance, the performance index used in previous studies [4, 5, 90] for the quantizer described in Equation (2) is defined as follows

$$E(Q) := \sup_{\substack{u \in U \\ k \in \{1,2,\ldots,L\}}} \operatorname{abs}\left(y_q(k) - y(k)\right), \tag{3}$$

where $L \in \mathbb{N}$ is the upper limit of the evaluation interval. For simplicity in this

study $L$ will be referred to just as the *evaluation interval*. $E(Q)$ gives the biggest possible value of $e(k)$, i.e., the worst case performance of the system.

This definition is very convenient. In [4] it was found that for linear plants, represented by Equation (1), the performance index is independent of the input signal and it is given by

$$E(Q) = \frac{d}{2} \sum_{k=0}^{L} \text{abs} \left( \begin{bmatrix} \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B}\mathcal{C} \\ \mathbf{0} & \mathcal{A} + \mathcal{B}\mathcal{C} \end{bmatrix}^k \begin{bmatrix} \mathbf{B} \\ \mathcal{B} \end{bmatrix} \right), \tag{4}$$

where the evaluation interval $L \to \infty$. However, for practical purposes $L$ can be considered as a finite big number. This performance index is used in this thesis for the design of finite level-dynamic quantizers and the design of event-triggered quantizers. In the case of neural network quantizers the performance index is based on the sum of square errors as it is done to train neural networks in the machine learning field. This index is defined as follows

$$E(Q) := \sum_{k=0}^{L} \left( y_q(k) - y(k) \right)^2. \tag{5}$$

In order to obtain the smallest performance degradation, it is necessary to make $E(Q)$ as small as possible by the appropriately designing the noise-shaping quantizer $Q$. Then, the quantizer design is reduced to an optimization problem, where the cost function is given by $E(Q)$ in Equation (3) and the design variables are the parameters of the quantizer $Q$.

### 2.1.2 Data Rate Constraints Management

The data rate constraint of the communication channel imposes limitations in the design of noise-shaping quantizers. Considering that $N_b$ is the number of bits that can be transmitted through the channel per sampling time, the number of quantization levels $M$ should satisfy the following condition

$$M \leq 2^{N_b}. \tag{6}$$

In this thesis it is assumed that $M$ is given satisfying Equation (6).

If there is no saturation in a finite-level static quantizer, then the maximum quantization error is given by $\Delta q = d/2$. Thus, in order to minimize $E(Q)$ it is

necessary to make $d$ as small as possible. However, the following condition should be satisfied to avoid saturation

$$\text{abs}\left(\bar{u}(k)\right) \leq \frac{1}{2}Md, \tag{7}$$

$\bar{u}(k)$ being the output of the digital filter. Then, since $M$ is limited by the data rate constraints, the minimum value of $d$ is compromised. For instance, in the case of a linear plant and the quantizer given by Equation (2), Okajima et al. in [90] found that the minimum $d$ that satisfies this condition is given by

$$d^* = \frac{(u_{max} - u_{min})}{M - \dfrac{\text{abs}(\boldsymbol{\mathcal{C}}\mathbf{T})\text{abs}(\mathbf{T}^{-1}\boldsymbol{\mathcal{B}})\bar{\Lambda}^L}{1-\bar{\Lambda}} - \displaystyle\sum_{k=0}^{L} \text{abs}\left(\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}})^k\boldsymbol{\mathcal{B}}\right)}, \tag{8}$$

where $\bar{\Lambda}$ is the maximum absolute value of the eigenvalues of $\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}}$ and $\mathbf{T}$ is the matrix composed of the right eigenvectors of $\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}}$ as columns that is used for diagonalization.

Thus, $d^*$ is used in this thesis to find the optimal finite-level dynamic quantizers and the event-triggered quantizer both subjected to data rate constraints. For the case of the neural network quantizer, there is not such expression, and the static quantizer may saturate. The difference between $d^*$ and $d$ is that $d^*$ represents the minimum value of $d$ for which the quantizer $Q$ does not saturate. The value of $d^*$ is specified in Equation (8) for the considered quantizer described in Equation (2). Different types of quantizers may have other expressions to evaluate $d^*$.

## 2.2 Design Problem and Constraints

The design of the considered noise-shaping quantizers are formulated as optimization problems. In these problems the performance index $E(Q)$ is minimized by the appropriate selection of the quantizer parameters.

These optimization problems happen to be nonlinear and nonconvex. Then, they cannot be solved directly by conventional optimization methods like linear programming or quadratic programming. Thus, alternative optimization methods should be used. In this thesis, the design problems are solved using metaheuristics. The metaheuristics considered in this study are the covariance matrix

adaptation evolution strategy (CMA-ES) [40, 37], differential evolution (DE) [123] and firefly algorithm (FA) [137]. These are all state of the art metaheuristics that have many appealing properties. They are described in detail in the next section.

The noise-shaping quantizers design problems are subjected to constraint conditions that define their space solutions. These constraints are different for the different types of quantizers considered. The metaheuristic algorithms used in this thesis were designed to solve unconstrained problems. Then, in order to manage the optimization problems constraints a method developed by Maruta et al. in [74] and [73] is employed. This method transforms the constrained optimization problem into an unconstrained one as it is explained below.

First, the optimization problem subjected to multiple constraint conditions should be formulated as follows

$$\underset{\boldsymbol{\theta}\in\mathbb{F}}{\text{minimize}}\, J_{org}(\boldsymbol{\theta}), \qquad \mathbb{F} := \{\boldsymbol{\theta} \mid p_1(\boldsymbol{\theta}) < 0, p_2(\boldsymbol{\theta}) < 0, \ldots, p_m(\boldsymbol{\theta}) < 0\}, \tag{9}$$

where $J_{org} : \mathbb{R}^n \to \mathbb{R}$ represents the original cost function, $\boldsymbol{\theta} \in \mathbb{R}^n$ is the design variable, $p_i : \mathbb{R}^n \to \mathbb{R}$ is a linear/nonlinear hard constraint function and $\mathbb{F}$ represents the feasible region, i.e., the set of solutions satisfying all constraint conditions. It is assumed that $\mathbb{F}$ is not empty.

Second, it is necessary to find a function $J_v : \mathbb{R}^n \to \mathbb{R}$ satisfying the following conditions:

(C1) $J_v(\boldsymbol{\theta}) < 0$ holds for any $\boldsymbol{\theta} \in \mathbb{F}$.

(C2) $J_v(\boldsymbol{\theta}_a) < J_v(\boldsymbol{\theta}_b)$ holds whenever $J_{org}(\boldsymbol{\theta}_a) < J_{org}(\boldsymbol{\theta}_b)$ is satisfied.

The function $J_v(\boldsymbol{\theta})$ is known as *virtual cost function* and it always exists. Notice that if $J_{org}(\boldsymbol{\theta})$ already satisfies the first condition (C1), the virtual cost can be simply set as $J_v(\boldsymbol{\theta}) = J_{org}(\boldsymbol{\theta})$.

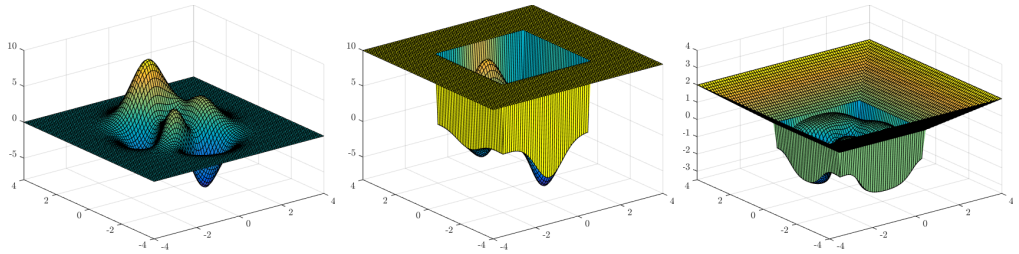
Lastly, once that $J_v(\boldsymbol{\theta})$ is selected, the constrained optimization problem in Equation (9) is transformed into the following unconstrained one

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}}\, J(\boldsymbol{\theta}), \qquad \text{for} \qquad J(\boldsymbol{\theta}) := \begin{cases} J_v(\boldsymbol{\theta}) & \text{if } p(\boldsymbol{\theta}) < 0, \\ p(\boldsymbol{\theta}) & \text{otherwise,} \end{cases} \tag{10}$$

where

$$p(\boldsymbol{\theta}) := \max[p_1(\boldsymbol{\theta}), p_2(\boldsymbol{\theta}), \ldots, p_m(\boldsymbol{\theta})]. \tag{11}$$

The function $p(\boldsymbol{\theta})$ is known as *penalty cost*, because this function is applied when the constraints of the problem are not satisfied. Then, the constraints of the problem act as variable penalty costs. A fixed penalty cost does not provide information about how far a candidate solution is out from $\mathbb{F}$, and it usually happens that when all the candidate solutions are initially out of the feasible region there is no way for the metaheuristic to make them go inside $\mathbb{F}$. The method described above solves this problem by providing the information about how far a possible solution is out of $\mathbb{F}$, this information is given by $p(\boldsymbol{\theta})$ in Equation (125). Figure 8 shows an example a fixed penalty cost method and a variable one. These penalty costs are applied the Peaks function of Matlab in the example.



(a) Cost function $J(\boldsymbol{\theta})$. (b) Fixed penalty $p$. (c) Variable penalty $p(\boldsymbol{\theta})$.

Figure 8: Penalty cost comparison. In this example the cost function is the Matlab's Peaks function and the constraints are the following: $-2 \leq \theta_1 \leq 2$, $-2 \leq \theta_2 \leq 2$.

## 2.3 Metaheuristics

Metaheuristics are high level strategies, often nature-inspired, for exploring feasible solutions to optimization problems. The metaheuristics have several advantages: they do not make assumptions on the problem to be solved; they do not need the gradient or Hessian matrix of the function to be optimized, and many of them are easy to implement and computationally inexpensive. As expected, they have some drawbacks as well such as there is not guarantee that an optimal solution is ever found, some are very sensitive to the tuning of their hyperparameters

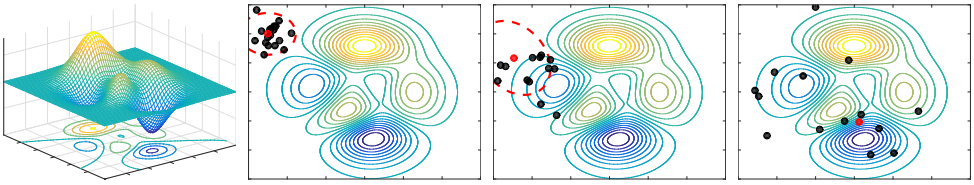
and the possibility of getting trapped into local minima [8, 138, 10].

The metaheuristics used for the quantizers design are covariance matrix adaptation evolution strategy (CMA-ES), differential evolution (DE) and firefly algorithm (FA). The main reason to chose these metaheuristics is that they show a very good performance in the optimization of multimodal functions with local minima. Additionally, they are easy to implement and require very few hyperparameters to be tuned. The candidate solutions are known in different ways in each of these metaheuristics. For instance, in CMA-ES they are called search points, and in DE they are known as target vectors. Then, to unify the nomenclature and to avoid confusion, in this thesis the possible solutions will be known as *individuals*. A brief description of these metaheuristics and their algorithms is presented below.

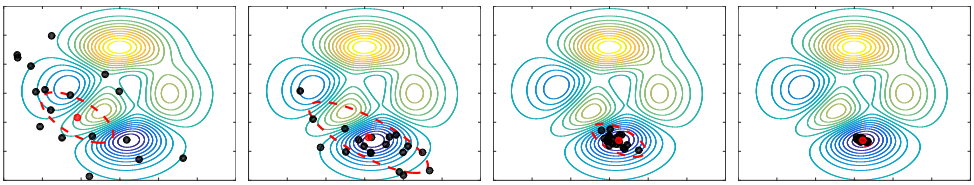
### 2.3.1 Covariance Matrix Adaptation Evolutionary Strategy

The covariance matrix adaptation evolution strategy (CMA-ES) is an evolutionary algorithm used for solving non-linear non-convex black-box optimization problems in continuous domains. It is considered as a state-of-the-art in evolutionary computation. In CMA-ES the possible solutions are generated randomly according to a multivariate normal distribution with mean $\boldsymbol{m}$ and covariance matrix $\boldsymbol{\Sigma}$. The initial value of $\boldsymbol{m}$ is provided by the user or it can be selected randomly inside the search space and initially $\boldsymbol{\Sigma} = \mathbf{I}$. Then, in each iteration of the algorithm, the best individuals are selected and the parameters of the normal distribution are updated. Thus, the mean $\boldsymbol{m}$ goes toward the best solution. In the next iteration the individuals are generated randomly according to the normal distribution with the new parameters [40, 39, 125]. An example of the operation of CMA-ES is shown in Figure 9 for a two dimensional optimization problem.

The main advantage of CMA-ES over the other metaheuristics is that it does not require the tunning of almost any parameter. The only parameter left to the user is the number of individuals $N$. By increasing $N$ from its default value, the exploration capabilities and robustness of CMA-ES are usually improved, while the convergence time increases. Other advantages are that it has several invariance properties and it shows a very good performance in solving non-separable and ill-conditioned problems.



(a) Peaks function. (b) Generation 1. (c) Generation 3. (d) Generation 6.



(e) Generation 9. (f) Generation 12. (g) Generation 15. (h) Generation 18.

Figure 9: Example of the operation of CMA-ES over a two dimensional search space. It is shown how the $N = 20$ individuals (black dots) and the mean (red dot) move in the search space through the generations. The red break line helps to see the adaptation of the covariance matrix, it represents a contour of the multivariate normal distribution with constant probability $p_1 = 0.1$.

This algorithm has been evolving since its first development around 1995. The version applied in this study to the dynamic quantizer design is the $(\mu/\mu_W, \lambda)$ CMA-ES strategy, described in [37]. In the $(\mu/\mu_W, \lambda)$ strategy, $\mu$ is the number of parents of the next generation, $\mu_W$ indicates a weighted recombination of the parents and $\lambda$ is the number of individuals, although, in this thesis $\lambda$ is represented by $N$ for comparison purposes with the other algorithms. The $(\mu/\mu_W, \lambda)$ CMA-ES algorithm is shown in Algorithm 1.

**Algorithm 1 : $(\mu/\mu_W, \lambda)$ CMA-ES**

**Initialization:** Given $N \in \mathbb{N}$, $k_{max} \in \mathbb{N}$, $\boldsymbol{m} \in \mathbb{R}^n$, the step size $\sigma \in \mathbb{R}_+$ and the initial search space $S_0 = [\theta_{min}, \theta_{max}]^n$. Initialize $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$, $\boldsymbol{p}_\sigma \in \mathbb{R}^n$ and $\boldsymbol{p}_c \in \mathbb{R}^n$ as $\boldsymbol{\Sigma} = \mathbf{I}$, $\boldsymbol{p}_\sigma = \mathbf{0}$ and $\boldsymbol{p}_c = \mathbf{0}$ respectively. Set the values of the parameters $c_c$, $c_\sigma$, $c_1$, $c_\mu$, $d_\sigma$, $\mu_{\text{eff}}$ and $w_i$ $(i = 1, 2, \ldots N)$ to their default values given in Appendix A. Then, $k = 0$.

**Step 1 (Sample new population):** $N$ individuals $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ are

generated randomly from the multivariable normal distribution $\mathcal{N}(\boldsymbol{m}, \sigma^2\boldsymbol{\Sigma})$ as follow

$$\boldsymbol{\theta}_i \;\; = \boldsymbol{m} + \sigma \boldsymbol{y}_i, \quad \boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \quad \text{for } i = 1, 2, \ldots, N \; . \tag{12}$$

**Step 2 (Selection and recombination):** The cost function $J(\boldsymbol{\theta}_i)$ is evaluated for each $\boldsymbol{\theta}_i$, then the individuals $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ and their respective $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\}$ are ordered based on the fitness value of $\boldsymbol{\theta}_i$. The ones with the best fitness go at the beginning.

The first $\mu$ individuals are the parents of the next generation. They are combined with each other to generate the new mean as follows

$$\boldsymbol{m} \;\; = \;\; \sum_{i=1}^{\mu} w_i \boldsymbol{\theta}_i = \boldsymbol{m} + \sigma \boldsymbol{y}_w, \tag{13}$$

$$\boldsymbol{y}_w \;\; = \;\; \sum_{i=1}^{\mu} w_i \boldsymbol{y}_i \; . \tag{14}$$

**Step 3 (Step size control):**

$$\boldsymbol{p}_\sigma \;\; \leftarrow \;\; (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{y}_w, \tag{15}$$

$$\sigma \;\; \leftarrow \;\; \sigma \times \exp\left[\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\boldsymbol{0}, \mathbf{I})\|} - 1\right)\right], \tag{16}$$

where $\mathsf{E}\|\mathcal{N}(\boldsymbol{0}, \mathbf{I})\| \approx \sqrt{n}\,(1 - 1/4n + 1/21n^2)$.

**Step 4 (Covariance matrix adaptation)**:

$$h_\sigma = \begin{cases} 1 & \text{if} \frac{\|\boldsymbol{p}_\sigma\|}{\sqrt{1-(1-c_\sigma)^{2(k+1)}}} < \left(1.5 + \frac{1}{n-0.5}\right) \mathsf{E}\|\mathcal{N}(\boldsymbol{0}, \mathbf{I})\|, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

$$\boldsymbol{p}_c \leftarrow (1 - c_c)\boldsymbol{p}_c + h_\sigma\sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\boldsymbol{y}_w, \tag{18}$$

$$\boldsymbol{\Sigma} \leftarrow (1 - c_1 - c_\mu)\boldsymbol{\Sigma} + c_1\left(\boldsymbol{p}_c\boldsymbol{p}_c^{\mathrm{T}} + (1 - h_\sigma)c_c(2 - c_c)\boldsymbol{\Sigma}\right) + c_\mu\sum_{i=1}^{\mu} w_i\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}} \; . \tag{19}$$

**Step 5 (Check stop condition):** If the stop condition is not satisfied $k \leftarrow k + 1$ and go to **Step 1**. Otherwise, terminate the algorithm and return $\boldsymbol{m}$ (or $\boldsymbol{\theta}_1$).

The default values of the CMA-ES' control parameters are shown in Appendix A. Meanwhile, the initial values of $\boldsymbol{m}$ and $\sigma$ are problem dependent and should be chosen appropriately. A criteria for the selection of the initial $\boldsymbol{m}$ is to sample it uniformly within a region of the search space where the user expects to find the global optima $S_0 = [\theta_{min}, \theta_{max}]^n$. If the global optima is not inside $S_0$, the CMA-ES algorithm will still be able to find it, but the convergence speed will be slower and the probability of the algorithm to get trapped into a local minimum will increase. On the other hand, the initial step size $\sigma$ should not be too small since it will reduce the algorithm performance on multimodal functions. A rule of the thumb empirically found is to make $\sigma = 0.3(\theta_{max} - \theta_{min})$.

The stop criteria of the CMA-ES algorithm is problem dependent too and it should be selected by the user. In this case the algorithm will terminate if the maximum number of iterations $k_{max}$ is reached or if the condition number of the covariance matrix $\boldsymbol{\Sigma}$ exceeds $10^{14}$. In order to compare the performance of CMA-ES against other metaheuristics different values of $k_{max}$ were used.

### 2.3.2 Differential Evolution

DE is a population based metaheuristic algorithm inspired in the mechanism of biological evolution [123, 100]. In this algorithm, the cost function $J(\boldsymbol{\theta})$ is evaluated iteratively over a population of individuals $\boldsymbol{\theta}_i$, known as target vectors in the DE literature, in each iteration the individuals improve their values and move towards the best solution. Finally the individual with the lowest fitness value in the last iteration is regarded as the optimal solution.

Some advantages of DE are that it is very easy to implement and has only two tuning parameters: the scale factor $F$ and the crossover constant $H$, apart from the number of individuals $N$ and the maximum number of iterations $k_{max}$. Besides, DE shows very good exploration capacities and converge very fast to the global optima. The DE algorithm is shown in Algorithm 2

---

**Algorithm 2 : DE** (DE/best/1/bin strategy)

---

**Initialization:** Given $N \in \mathbb{N}$, $k_{max} \in \mathbb{N}$, $F \in [0, 2]$, $H \in [0, 1]$ and the initial search space $S_0 = [\theta_{min}, \theta_{max}]^n$. Set $k = 0$ then select randomly $N$ individuals $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ in the search space.

**Step 1:** The cost function $J(\boldsymbol{\theta})$ is evaluated for each $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_{base} = \boldsymbol{\theta}_{l_k}$ is calculated by:

$$l_k \quad = \arg\min_{i\in\{1,2,\ldots,N\}} J(\boldsymbol{\theta}_i). \tag{20}$$

If $k = k_{max}$ then $\boldsymbol{\theta}_{base}$ is the final solution, if not go to **step 2**.

**Step 2 (Mutation):** For each $\boldsymbol{\theta}_i$ a mutant vector $\boldsymbol{\mathfrak{M}}_i$ is generated by:

$$\boldsymbol{\mathfrak{M}}_i = \boldsymbol{\theta}_{base} + F(\boldsymbol{\theta}_{\tau_{1,i}} - \boldsymbol{\theta}_{\tau_{2,i}}), \tag{21}$$

where $\tau_{1,i}$ and $\tau_{2,i}$ are random indexes subject to $i \neq \tau_{1,i} \neq \tau_{2,i} \neq l_k$.

**Step 3 (Crossover):** For each $\boldsymbol{\theta}_i$ and $\boldsymbol{\mathfrak{M}}_i$ a trial vector $\boldsymbol{\mathfrak{T}}_i$ is generated by:

$$\mathfrak{T}_{i,j} = \begin{cases} \mathfrak{M}_{i,j} & \text{if } \rho_{i,j} \leq H \text{ or } j = j_{rand}, \\ \theta_{i,j} & \text{otherwise,} \end{cases} \tag{22}$$

where $\rho_{i,j} \in [0,1]$ and $j_{rand} \in \{1,2,\ldots,n\}$ are generated randomly.

**Step 4 (Selection):** The members of the next generation $k+1$ are selected by:

$$\boldsymbol{\theta}_i \leftarrow \begin{cases} \boldsymbol{\mathfrak{T}}_i & \text{if } J(\boldsymbol{\mathfrak{T}}_i) \leq J(\boldsymbol{\theta}_i), \\ \boldsymbol{\theta}_i & \text{otherwise,} \end{cases} \tag{23}$$

then $k \leftarrow k+1$ and go to **step 1**.

---

### 2.3.3 Firefly Algorithm

The firefly algorithm is a population based metaheuristic that is inspired in the flashing behavior of tropical fireflies. It was first introduced in [137], and it has been evolving since then. The algorithm is based on the following three idealize rules about the fireflies behavior:

1. Each firefly will be attracted to other fireflies regardless of their sex.

2. The attractiveness between two fireflies is proportional to their brightness and it decreases as the distance between them increases. The less brighter firefly will move towards the brighter one and the brightest in the swarm will move randomly.

3. The brightness of a firefly is determined by the landscape of the cost function.

Some advantages of the FA algorithm over the other metaheuristics are the automatic subdivision of the set of individuals, the ability to deal with multimodality and the online tuning of the parameters to control the randomness in the update law [139].

The FA algorithm has the following parameters: the initial randomness scaling factor $\alpha_0$, the initial attractiveness $\beta_0$, the absorption coefficient $\gamma$ and the cooling factor $\delta$. The FA implementation used in this study is shown in Algorithm 3.

**Algorithm 3 : FA**

**Initialization:** Given $N \in \mathbb{N}$, $k_{max} \in \mathbb{N}$, $\alpha_0$, $\beta_0$, $\gamma \in [0, \infty)$ (in practice $[0.1, 10]$), $\delta \in (0, 1)$ and the initial search space $S_0 = [\theta_{min}, \theta_{max}]^n$. Set $k = 0$, then select randomly $N$ individuals $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ within $S_0$.

**Step 1:** The cost function $J(\boldsymbol{\theta}_i)$ is evaluated for each $\boldsymbol{\theta}_i$, then the individuals are sorted in ascending order based on their fitness value. If $k = k_{max}$ then $\boldsymbol{\theta}_N$ is the solution of the algorithm, if not go to **Step 2**.

**Step 2:** Find the limits of the search space. Considering that $\boldsymbol{L}_b = [l_1, l_2, \ldots, l_n]$ and $\boldsymbol{U}_b = [u_1, u_2, \ldots, u_n]$, then for $j = \{1, 2, \ldots, n\}$:

$$l_j = \min_{\theta_j \in \{\theta_{i,j} | i=1,2,\ldots,N\}} \theta_j, \tag{24}$$

$$u_j = \max_{\theta_j \in \{\theta_{i,j} | i=1,2,\ldots,N\}} \theta_j, \tag{25}$$

where $\theta_{i,j}$ is the $j^{th}$ component of the individual $\boldsymbol{\theta}_i$.

**Step 3:** Reduce the randomness:

$$\alpha = \alpha_0 \delta^k, \tag{26}$$

**Step 4:** For $i = \{1, 2, \ldots, (N-1)\}$ the following update law is applied to each individual:

$$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \sum_{j=i+1}^{N} \left[\beta_0 \left(\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\right) \exp\left(-\gamma \left\|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\right\|_2^2\right) + \alpha \boldsymbol{\epsilon}_j\right], \tag{27}$$

$$\boldsymbol{\epsilon}_j = \rho_j (\boldsymbol{U}_b - \boldsymbol{L}_b), \tag{28}$$

where $\rho_j \in [-0.5, 0.5]$ is a random number uniformly distributed. Then, make $k \leftarrow k + 1$ and go to **Step 1**.

# 3. Design of Finite-level Quantizers Under Data Rate Constraints

In this chapter the design of finite-level dynamic quantizers affected by the channel's data rate constraints is studied. With numerical examples it is verified that the quantizers and the proposed design methods work properly. After that, the methods based on different metaheuristics are compared among each other and with previously developed methods in order to evaluate their performance.

## 3.1 Considered System

The finite-level noise-shaping dynamic quantizer is designed using the feedforward error system depicted in Figure 6. The assumptions made for the design of this quantizers are the following: the communication channel has no losses and no delays, the plant $P$ is stable and the input signal is bounded, i.e., $u(k) \in U$ for $U := [u_{min}, u_{max}]$. The considered plant $P$ is the one represented in Equation (1) which is a discrete-time and single-input-single-output (SISO) system. This plant is assumed to be stable. This assumption implies that all the eigenvalues of the matrix $\mathbf{A}$ are inside the unit circle in the complex plane. The dynamic quantizer $Q$ considered in this chapter is the one described in Equation (2). The digital filter and the static quantizer q[·] are the same. In this chapter it is assumed that $M$ is given by the designer. Thus, the design parameters of the dynamic quantizer $Q$ are $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, $\boldsymbol{\mathcal{C}}$ and $d$.

An important care that should be taken into account when designing a dynamic quantizer is to make the quantizer stable. The stability condition for the quantizer can be derived easily. First, the quantizer's output $v(k)$ given in Equation (2) is represented as follows

$$v(k) = \boldsymbol{\mathcal{C}}\boldsymbol{\xi}(k) + u(k) + w(k), \tag{29}$$

where $w(k) \in \left[-\frac{d}{2}, \frac{d}{2}\right]$ is the *quantization error*. Then, substituting the new $v(k)$ into Equation (2) the following representation of the quantizer is obtained

$$Q : \begin{cases} \boldsymbol{\xi}(k+1) & = (\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}})\,\boldsymbol{\xi}(k) + \boldsymbol{\mathcal{B}}w(k), \\ v(k) & = \boldsymbol{\mathcal{C}}\boldsymbol{\xi}(k) + w(k) + u(k). \end{cases} \tag{30}$$

In this representation the quantizer's state $\boldsymbol{\xi}(k)$ depends only on the quantization error $w(k)$ which works as a bounded signal. Thus, the quantizer $Q$ is bounded-input-bounded-output (BIBO) stable if all the eigenvalues of $\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}}$ are inside the unit circle in the complex plane. This condition can be expressed as

$$\bar{\Lambda} = \max_i \operatorname{abs}\left(\Lambda_i(\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}})\right) < 1, \tag{31}$$

where $\bar{\Lambda}$ is the maximum absolute value of the eigenvalues of the $\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}}$. The BIBO stability for LTI systems establishes that every bounded input signal excites a bounded output signal [15]. Formally, a discrete-time SISO system is BIBO stable if and only if the system's impulse response $g(k)$ is absolutely summable in the interval $[0, \infty)$ or

$$\sum_{k=0}^{\infty} \operatorname{abs}(g(k)) \leq K < \infty \tag{32}$$

for some constant K.

## 3.2 Performance Index

As it was mentioned in Section 2.1.1, a performance index is used as a measurement of the system's performance degradation. In particular, the performance index considered in this chapter is the one defined in Equation (3).

For the considered system this index takes the form given in Equation (4). This expression was found by Azuma and Sugie in [4], and its development is carried out as follows. Putting together Equations (1) and (2) gives

$$\begin{cases} \begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{\xi}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & 0 \\ 0 & \boldsymbol{\mathcal{A}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{\xi}(k) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \boldsymbol{\mathcal{B}} \end{bmatrix} v(k) - \begin{bmatrix} 0 \\ \boldsymbol{\mathcal{B}} \end{bmatrix} u(k), \\ \\ y(k) = \begin{bmatrix} \mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{\xi}(k) \end{bmatrix}. \end{cases} \tag{33}$$

By replacing the quantized output $v(k)$ with Equation (29), the Equation (33) can be expressed as:

$$\begin{cases} \bar{\boldsymbol{x}}(k+1) = \bar{\mathbf{A}}\bar{\boldsymbol{x}}(k) + \bar{\mathbf{B}}_1 w(k) + \bar{\mathbf{B}}_2 u(k), \\ y(k) = \bar{\mathbf{C}}\bar{\boldsymbol{x}}(k), \end{cases} \tag{34}$$

where:

$$\bar{\boldsymbol{x}}(k) = \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{\xi}(k) \end{bmatrix}, \bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{B}\mathcal{C} \\ 0 & \mathcal{A} + \mathcal{B}\mathcal{C} \end{bmatrix}, \bar{\mathbf{B}}_1 = \begin{bmatrix} \mathbf{B} \\ \mathcal{B} \end{bmatrix}, \bar{\mathbf{B}}_2 = \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix}, \bar{\mathbf{C}} = \begin{bmatrix} \mathbf{C} & 0 \end{bmatrix}. \tag{35}$$

The expression of the state $\bar{\boldsymbol{x}}(k)$ for this system is given as follows

$$\bar{\boldsymbol{x}}(k) = \bar{\mathbf{A}}^k \bar{\boldsymbol{x}}(0) + \sum_{i=0}^{k-1} \bar{\mathbf{A}}^{k-1-i} \left( \bar{\mathbf{B}}_1 w(i) + \bar{\mathbf{B}}_2 u(i) \right). \tag{36}$$

Then, Equation (36) is taken into Equation (34) in order to find the plant output under the effects of quantization, the real output $y_q(k)$. The result is

$$y_q(k) = \bar{\mathbf{C}}\bar{\mathbf{A}}^k \bar{\boldsymbol{x}}(0) + \sum_{i=0}^{k-1} \bar{\mathbf{C}}\bar{\mathbf{A}}^{k-1-i} \left( \bar{\mathbf{B}}_1 w(i) + \bar{\mathbf{B}}_2 u(i) \right). \tag{37}$$

Considering the value $\bar{\mathbf{A}}$ in Equation (35), the expresion $\bar{\mathbf{A}}^k$ can be reduced to

$$\bar{\mathbf{A}}^k = \begin{bmatrix} \mathbf{A}^k & \Omega(k) \\ \mathbf{0} & (\mathcal{A} + \mathcal{B}\mathcal{C})^k \end{bmatrix}, \tag{38}$$

where $\Omega(k)$ is an expression that increases with $k$, in this case its value is not important since it will disappear from the calculation. Effectively, remembering that $\bar{\boldsymbol{x}}(0) = [\boldsymbol{x}(0) \quad \boldsymbol{\xi}(0)]^\top = [\boldsymbol{x}(0) \quad 0]^\top$, the terms of Equation (37) are represented by

$$\bar{\mathbf{C}}\bar{\mathbf{A}}^k \bar{\boldsymbol{x}}(0) = \mathbf{C}\mathbf{A}^k \boldsymbol{x}(0), \tag{39}$$

$$\bar{\mathbf{C}}\bar{\mathbf{A}}^{k-1-i} \bar{\mathbf{B}}_2 u(i) = \mathbf{C}\mathbf{A}^{k-1-i} \mathbf{B} u(i). \tag{40}$$

Then, $y_q(k)$ can be rewritten as

$$y_q(k) = \mathbf{C}\mathbf{A}^k \boldsymbol{x}(0) + \sum_{i=0}^{k-1} \bar{\mathbf{C}}\bar{\mathbf{A}}^{k-1-i} \bar{\mathbf{B}}_1 w(i) + \sum_{i=0}^{k-1} \mathbf{C}\mathbf{A}^{k-1-i} \mathbf{B} u(i). \tag{41}$$

For the case in which the control signal $u(k)$ is applied directly to the plant $P$ (ideal case) the plant's output $y(k)$ is

$$y(k) = \mathbf{C}\mathbf{A}^k \boldsymbol{x}(0) + \sum_{i=0}^{k-1} \mathbf{C}\mathbf{A}^{k-1-i} \mathbf{B} u(i). \tag{42}$$

The difference between the Equations (41) and (42) gives

$$y_q(k) - y(k) = \sum_{i=0}^{k-1} \bar{\mathbf{C}}\bar{\mathbf{A}}^{k-1-i}\bar{\mathbf{B}}_1 w(i). \tag{43}$$

Taking the absolute value of both members of Equation (43) and considering that the quantized error $\text{abs}\,(w(i)) \leq \frac{d}{2}$ produce the following

$$\begin{aligned} \text{abs}\,(y_q(k) - y(k)) &= \text{abs}\left(\sum_{i=0}^{k-1} \bar{\mathbf{C}}\bar{\mathbf{A}}^{k-1-i}\bar{\mathbf{B}}_1 w(i)\right), \\ &\leq \sum_{i=0}^{k-1} \text{abs}\left(\bar{\mathbf{C}}\bar{\mathbf{A}}^{k-1-i}\bar{\mathbf{B}}_1\right) \text{abs}\,(w(i)), \\ &\leq \frac{d}{2}\sum_{i=0}^{k-1} \text{abs}\left(\bar{\mathbf{C}}\bar{\mathbf{A}}^{k-1-i}\bar{\mathbf{B}}_1\right). \end{aligned} \tag{44}$$

By replacing the values in Equation (35) to Equation (44) the next expression is obtained

$$\text{abs}\,(y_q(k) - y(k)) \leq \frac{d}{2}\sum_{i=0}^{k-1} \text{abs}\left(\begin{bmatrix}\mathbf{C} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{A} & \mathbf{B}\boldsymbol{\mathcal{C}} \\ \mathbf{0} & \boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}}\end{bmatrix}^{k-1-i}\begin{bmatrix}\mathbf{B} \\ \boldsymbol{\mathcal{B}}\end{bmatrix}\right). \tag{45}$$

Finally, in [4] it is proven that the equality in Equation (45) gives the supremum of $\text{abs}\,(y_q(k) - y(k))$, this means that the performance index is given by Equation (4).

## 3.3 Minimum Quantization Interval

As mentioned in Section 2.1.2 the data rate constraints of the channel pose some limitations in the design of finite-level dynamic quantizer. In this study it is assumed that the number of quantization levels $M$ is given by the designer satisfying the condition in Equation (6).

On the other hand, Equation (4) states that $d$ is directly proportional to $E(Q)$. Then, in order to minimize $E(Q)$ it is necessary to make $d$ as small as possible. However, $d$ and $M$ are subjected to the following condition derived from Figure 7

$$\text{abs}\,(\boldsymbol{\mathcal{C}}\boldsymbol{\xi}(k) + u(k)) \leq \frac{1}{2}Md, \tag{46}$$

this is the same condition as Equation (7).

The minimization of $d$ under this condition is equivalent to a reachable set problem not easy to solve. Okajima et al. in [90], [91] and [92] found an expression for the minimum quantization interval $d^*$ that satisfies the data rate constraints, this expression is the one in Equation (8).

The development of that expression is carried out as follow. In order to design $d$, it is necessary to consider the values of $\boldsymbol{\xi}(k)$ and $u(k)$. Although the range of $u(k)$ is given by $U = [u_{min}, u_{max}]$ the range of $\boldsymbol{\xi}(k)$ is unknown. Thus, the range of $\boldsymbol{\xi}(k)$ should be estimated for a given $Q$.

The range of the quantizer $\bar{U} = [\bar{u}_{min}, \bar{u}_{max}]$ can be characterize by the ranges of $u(k)$ and $\psi = \mathcal{C}\boldsymbol{\xi}$ as follows

$$\bar{U} = [u_{min} + \psi_{min}, u_{max} + \psi_{max}]\,. \tag{47}$$

Considering Equation (29) a normalized version of the quantization error $w(k)$ is given by

$$\bar{w}(k) = \frac{w(k)}{d/2} = \frac{2}{d}\left(v(k) - \mathcal{C}\boldsymbol{\xi} - u(k)\right), \tag{48}$$

where $\bar{w}(k) \in [-1, 1]$ and, in consequence, $|\bar{w}(k)| \leq 1$. From this normalized error the following system is built

$$H : \begin{cases} \boldsymbol{\xi}(k+1) & = (\mathcal{A} + \mathcal{B}\mathcal{C})\,\boldsymbol{\xi}(k) + \frac{d}{2}\mathcal{B}\bar{w}(k), \\ \psi(k) & = \mathcal{C}\boldsymbol{\xi}(k), \qquad |\bar{w}(k)| \leq 1, \end{cases} \tag{49}$$

by making the following change in the coordinates $\boldsymbol{\xi} = \frac{d}{2}\bar{\boldsymbol{\xi}}$ the system $H$ can be represented as

$$\bar{H} : \begin{cases} \bar{\boldsymbol{\xi}}(k+1) & = (\mathcal{A} + \mathcal{B}\mathcal{C})\,\bar{\boldsymbol{\xi}}(k) + \mathcal{B}\bar{w}(k), \\ \bar{\psi}(k) & = \mathcal{C}\bar{\boldsymbol{\xi}}(k), \qquad |\bar{w}(k)| \leq 1, \end{cases} \tag{50}$$

where $\bar{\psi} = 2\psi/d$. Notice that the system $\bar{H}$ is independent of $d$.

The reachable set problem is reduced into finding the $\bar{\psi}_{min}$ and $\bar{\psi}_{max}$ subjected to the condition that

$$\bar{\psi}_{min} \leq \mathcal{C}\bar{\boldsymbol{\xi}}(k) \leq \bar{\psi}_{max}, \qquad \forall \bar{\boldsymbol{\xi}}(k) \in \Xi, \tag{51}$$

where $\Xi$ is the reachable set of $\bar{\boldsymbol{\xi}}(k)$ to $\bar{w}(k)$.

From the Equations (7) and (47) the following conditions can be derived

$$\begin{aligned}\mathcal{C}\boldsymbol{\xi}(k)+u(k) \leq \quad & u_{max}+\psi_{max} \quad = u_{max}+\frac{d}{2}\bar{\psi}_{max},\\ \mathcal{C}\boldsymbol{\xi}(k)+u(k) \geq \quad & u_{min}+\psi_{min} \quad = u_{min}+\frac{d}{2}\bar{\psi}_{min},\end{aligned}$$

and the range of the quantizer is subjected to the following condition

$$\begin{aligned}\bar{u}_{max}-\bar{u}_{min} &\leq Md,\\ \left(u_{max}+\frac{d}{2}\bar{\psi}_{max}\right)-\left(u_{min}+\frac{d}{2}\bar{\psi}_{min}\right) &\leq Md,\\ (u_{max}-u_{min})+\frac{d}{2}\left(\bar{\psi}_{max}-\bar{\psi}_{min}\right) &\leq Md,\\ \frac{(u_{max}-u_{min})}{M-\frac{1}{2}\left(\bar{\psi}_{max}-\bar{\psi}_{min}\right)} &\leq d. \end{aligned} \tag{52}$$

By using the optimal values of $\bar{\psi}_{min}$ and $\bar{\psi}_{max}$ the minimium quantization interval $d^{opt}$ is given by

$$d^{opt}=\frac{(u_{max}-u_{min})}{M-\frac{1}{2}\left(\bar{\psi}^{opt}_{max}-\bar{\psi}^{opt}_{min}\right)}, \tag{53}$$

as it is shown in [90]. If $M-\frac{1}{2}\left(\bar{\psi}^{opt}_{max}-\bar{\psi}^{opt}_{min}\right)\leq 0$, there is no $d$ that satisfies the data rate constraints and the quantizer should be redesign.

Considering the system in Equation (50), the value of $\bar{\psi}(k)$ can be obtained by evaluating the system from $\tau=0,1,2,\ldots,k-1$ and having in account that $\bar{\boldsymbol{\xi}}(0)=\frac{2}{d}\boldsymbol{\xi}(0)=0$. The value of $\bar{\psi}(k)$ is given by

$$\bar{\psi}(k)=\sum_{\tau=0}^{k-1}\mathcal{C}(\mathcal{A}+\mathcal{B}\mathcal{C})^{k-1-\tau}\mathcal{B}\bar{w}(\tau), \tag{54}$$

and the reachable set boundaries are given as follow

$$\bar{\psi}^{opt}_{min} = -\sum_{k=0}^{\infty}\text{abs}\left(\mathcal{C}(\mathcal{A}+\mathcal{B}\mathcal{C})^{k}\mathcal{B}\right), \tag{55}$$

$$\bar{\psi}^{opt}_{max} = +\sum_{k=0}^{\infty}\text{abs}\left(\mathcal{C}(\mathcal{A}+\mathcal{B}\mathcal{C})^{k}\mathcal{B}\right). \tag{56}$$

In reality the series in Equations (55) and (56) cannot be evaluated until $\infty$. Thus, the values of $\bar{\psi}^{opt}_{min}$ and $\bar{\psi}^{opt}_{min}$ are approximated by evaluating the equations until a finite number $L$, the *evaluation interval*. This gives

$$\bar{\psi}^{L}_{min} = -\sum_{k=0}^{L} \text{abs}\left(\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}+\boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}})^{k}\boldsymbol{\mathcal{B}}\right), \tag{57}$$

$$\bar{\psi}^{L}_{max} = +\sum_{k=0}^{L} \text{abs}\left(\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}+\boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}})^{k}\boldsymbol{\mathcal{B}}\right). \tag{58}$$

Then, Okajima et al. in [91] defined $\hat{\psi}$ as follows

$$\begin{aligned}\hat{\psi} &:= \frac{\text{abs}(\boldsymbol{\mathcal{C}}\mathbf{T})\text{abs}(\mathbf{T}^{-1}\boldsymbol{\mathcal{B}})\bar{\Lambda}^{L}}{1-\bar{\Lambda}}, \\ &= \sum_{k=L}^{\infty} \text{abs}(\boldsymbol{\mathcal{C}}\mathbf{T})\text{abs}(\mathbf{T}^{-1}\boldsymbol{\mathcal{B}})\bar{\Lambda}^{k}, \\ &\geq \sum_{k=L}^{\infty} \text{abs}\left(\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}+\boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}})^{k}\boldsymbol{\mathcal{B}}\right),\end{aligned} \tag{59}$$

where $\bar{\Lambda}$ is given by Equation (31) and $\mathbf{T}$ is the matrix used for the diagonalization of $(\boldsymbol{\mathcal{A}}+\boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}})$, which means that its columns are the eigenvectors of $(\boldsymbol{\mathcal{A}}+\boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}})$. $\hat{\psi}$ satisfies these relations

$$\bar{\psi}^{L}_{min} - \hat{\psi} \leq \bar{\psi}^{opt}_{min} \leq \bar{\psi}^{L}_{min}, \tag{60}$$

$$\bar{\psi}^{L}_{max} \leq \bar{\psi}^{opt}_{max} \leq \bar{\psi}^{L}_{max} + \hat{\psi}, \tag{61}$$

Finally replacing $\bar{\psi}^{opt}_{min}$ and $\bar{\psi}^{opt}_{max}$ in Equation (53) for $\left(\bar{\psi}^{L}_{min} - \hat{\psi}\right)$ and $\left(\bar{\psi}^{L}_{max} + \hat{\psi}\right)$ respectively the expression below is produced

$$d^{*} = \frac{(u_{max}-u_{min})}{M - \hat{\psi} - \frac{1}{2}\left(\bar{\psi}^{L}_{max} - \bar{\psi}^{L}_{min}\right)}. \tag{62}$$

which is the approximated version of Equation (53). It is fair to notice that

$$d^{opt} = \lim_{L\to\infty} d^{*}. \tag{63}$$

Putting all the Equations (57), (58) and (59) into Equation (62) the expression in Equation (8) is obtained. The evaluation interval $L$ should be selected as a

big number to reduce the error due to the approximation. Therefore, $d^*$ is used for the design of the finite-level dynamic quantizer. Notice that the condition $d^* > 0$ should be satisfied when designing the quantizer. Since the numerator in Equation (62) is always bigger than zero, this condition depends only on the denominator being bigger than zero or not.

## 3.4 Design Problem

The finite-level dynamic quantizer design problem is formulated as follow:

> Suppose that the plant $P$, the number of quantization levels $M$ and the input signal range $U$ are given. Then, find the quantizer parameters $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, $\boldsymbol{\mathcal{C}}$ and $d$ which minimize $E(Q)$, under the conditions that:
>
> i $Q$ is stable ($\bar{\Lambda} < 1$), and
>
> ii The data rate constraint is satisfied ($d = d^* > 0$).

The optimization problem considered here is non-linear and non-convex, so it can not be directly solved by conventional optimization methods. For this reason a metaheuristic approach is used to approximately solve the problem.

This quantizer design problem is subjected to two constraints that define its space solution. The management of these constraints is carried out using the method described in Section 2.2. Then, it is necessary to represent the constrained optimization problem in the form shown in Equation (9). The original cost function is

$$J_{org}(\boldsymbol{\theta}) = E(\boldsymbol{\theta}) \quad | \quad d = d^*(\boldsymbol{\theta}), \tag{64}$$

where $E(\boldsymbol{\theta})$ is the performance index in Equation (4), $d^*$ is the quantization interval given by Equation (8) and the candidate solutions $\boldsymbol{\theta}$ are composed of the $n$ unknown elements of the matrices $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$. Then, the constraints of the problem take the form

$$\begin{array}{llll} \text{Stability constraint:} & \bar{\Lambda}(\boldsymbol{\theta}) < 1 & \Rightarrow & p_1(\boldsymbol{\theta}) = \bar{\Lambda}(\boldsymbol{\theta}) - 1, \\ \text{Data rate constraint:} & d^*(\boldsymbol{\theta}) > 0 & \Rightarrow & p_2(\boldsymbol{\theta}) = -d^*(\boldsymbol{\theta}), \end{array} \tag{65}$$

where $\bar{\Lambda}(\boldsymbol{\theta})$ is given by Equation (31).

There are many ways to select $J_v(\boldsymbol{\theta})$ satisfying the conditions (C1) and (C2). In this study the following virtual function is used

$$J_v(\boldsymbol{\theta}) = \arctan[J_{org}(\boldsymbol{\theta})] - \pi/2, \tag{66}$$

since it probed to be effective in the solution of the problems presented in [73].

Finally, the dynamic quantizer design problem is expressed by the unconstrained optimization problem given by

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}}\, J(\boldsymbol{\theta}), \qquad \text{for} \qquad J(\boldsymbol{\theta}) := \begin{cases} J_v(\boldsymbol{\theta}) & \text{if } p(\boldsymbol{\theta}) < 0, \\ p(\boldsymbol{\theta}) & \text{otherwise,} \end{cases} \tag{67}$$

$$J_v(\boldsymbol{\theta}) = \arctan[E(\boldsymbol{\theta})] - \pi/2 \quad \text{with} \quad d = d^*(\boldsymbol{\theta}), \tag{68}$$

$$p(\boldsymbol{\theta}) = \max[\bar{\Lambda}(\boldsymbol{\theta}) - 1,\ -d^*(\boldsymbol{\theta})]. \tag{69}$$

## 3.5 Design Variable Setting

The quantizer design method consists in finding the values of $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, $\boldsymbol{\mathcal{C}}$ and $d$ of the dynamic quantizer given in Equation (2) that solve the unconstrained optimization problem given by Equation (67). The optimization is carried out by using one of the metaheuristics described in Section 2.3 (CMA-ES, DE or FA). The cost function for the optimization is $J(\boldsymbol{\theta})$ in Equation (67) with $J_v(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta})$ given by Equations (68) and (69) respectively. In addition, the design variable $\boldsymbol{\theta} \in \mathbb{R}^n$ is constructed with the $n$ unknown elements of the matrices $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$.

Then, given the plant $P$ with order $n_p$, the first step is to choose the order and form of the quantizer. The selection of the quantizers' order can be subjected to the precision wanted or the computational capacity available to implement it.

Once the order of the quantizer is chosen, it is time to set the form of the quantizer. Of course, it is possible to consider all the elements of $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$ as variables for the optimization problem, in that case we will have $n = 2n_q + n_q^2$ variables in total. However, this setting is not optimal since it is possible to represent the same quantizer into its equivalent *canonical controllable* form. Thus, it will be better to express the quantizer directly into its canonical form because in that case the amount of variables will be less ($n = 2n_q$) and the metaheuristic

will found the solutions faster and more precisely. With this setting the form of the quantizer is given by

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_1 & \theta_2 & \theta_3 & \cdots & \theta_{\frac{n}{2}} \end{bmatrix}, \quad \boldsymbol{\mathcal{B}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \quad \boldsymbol{\mathcal{C}} = \begin{bmatrix} \theta_{\frac{n}{2}+1} & \theta_{\frac{n}{2}+2} & \cdots & \theta_n \end{bmatrix} \tag{70}$$

Thereby, the form of the design variable or individuals for the metaheuristic is $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_n\}$. Note that, although $d$ is regarded as a design parameter as well, its value is a function of $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$, since $d = d^*$ and $d^*$ is given by the Equation (8).

## 3.6 Numerical Examples

A series of simulations were performed with the purpose of verifying the effectiveness of the proposed metaheuristic based quantizer design method. The metaheuristics used are CMA-ES, DE, FA and PSO. The last one, PSO, was employed in a previous study [140] and it is used here for comparison purposes. Appendix B shows the PSO algorithm.

Considering the system in Figure 6, two different plants $P$ were selected. One is the second order system:

$$P_1(s) = \frac{s+20}{s^2+3s+2}, \tag{71}$$

and the other is the third order system:

$$P_2(s) = \frac{s+10}{s^3+6s^2+9s+10}. \tag{72}$$

These plants are represented by their *transfer functions* which are defined for LTI systems as the ratio of the Laplace transform of the output to the Laplace transform of the input under the assumption the initial conditions are zero [87, 26]. Where in general $s$ is a complex variable. After discretizing these plants with the sampling time $t_S = 0.1[s]$ the following matrices for their state space

representations in Equation (1) are obtained:

$$P_1: \ \mathbf{A}_1 = \begin{bmatrix} 1.7236 & -0.7408 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{B}_1 = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix},$$

$$\mathbf{C}_1 = \begin{bmatrix} 0.3533 & -0.0083 \end{bmatrix}, \tag{73}$$

$$P_2: \ \mathbf{A}_2 = \begin{bmatrix} 2.4775 & -1.0169 & 0.5488 \\ 2 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} 0.1250 \\ 0 \\ 0 \end{bmatrix},$$

$$\mathbf{C}_2 = \begin{bmatrix} 0.0443 & 0.0169 & -0.0184 \end{bmatrix}. \tag{74}$$

For both plants the parameters for the quantizer are: $U = [-1, 1]$, $M = 2$ and $L = 150$. For $P_1$ and $P_2$, the form of the quantizer constant matrices are given by

$$\boldsymbol{\mathcal{A}}_1 = \begin{bmatrix} 0 & 1 \\ \theta_1 & \theta_2 \end{bmatrix}, \quad \boldsymbol{\mathcal{B}}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{\mathcal{C}}_1 = \begin{bmatrix} \theta_3 & \theta_4 \end{bmatrix}, \tag{75}$$

$$\boldsymbol{\mathcal{A}}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \theta_1 & \theta_2 & \theta_3 \end{bmatrix}, \quad \boldsymbol{\mathcal{B}}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{\mathcal{C}}_2 = \begin{bmatrix} \theta_4 & \theta_5 & \theta_6 \end{bmatrix}, \tag{76}$$

respectively. Then, the form of the individuals for the metaheuristic algorithms are $\boldsymbol{\theta}^{(1)} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix}^\top$ for $P_1$ and $\boldsymbol{x}^{(2)} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 \end{bmatrix}^\top$ for $P_2$, which implies that the dimension of the optimization problems are $n_1 = 4$ and $n_2 = 6$, respectively.

The hyperparameters of the metaheuristics employed are given in the Table 1. The metaheuristics' performance is subjected to the value of the hyperparameters. In this case we chose the control parameters based on the recommended values of previous studies. For DE the same parameter as in [78] are used, for FA the recommendations in [139] are followed and for PSO, in order to compare the results, the same hyperparameters as in [140] are set. In the case of CMA-ES the hyperparameters are the initial values of $\sigma$ and $\boldsymbol{m}$ but they can be generated automatically from the initial search interval $S_0$ according to [37].

Table 1: Hyperparameters used in the simulations.

| CMA-ES | DE | FA | PSO |
|---|---|---|---|
| $\sigma_0 = 0.3$ | $F = 0.6$ | $\alpha_0 = 0.25$ | $\chi_0 = 0.9$ |
| | $H = 0.9$ | $\beta_0 = 1$ | $\chi_1 = 1$ |
| | | $\gamma = 0.1$ | $\chi_2 = 1$ |
| | | $\delta = 0.97$ | |

For DE, FA and PSO the initial individuals (members of the first generation) are generated randomly inside the initial search space $S_0 = [-1, 1]^n$ by an uniform probability distribution. Additionally, for PSO the initial velocities of the particles are randomly initialized in the range $V_0 = [0, 1]^n$. For CMA-ES the initial mean $\boldsymbol{m}_0$ is selected randomly using a uniform probability distribution within $S_0 = [-1, 1]^n$ and the initial individuals are sampled from the normal probability distribution $\mathcal{N}(\boldsymbol{m}, \sigma^2\mathbf{I})$.

The simulations are performed by trying $N_{run} = 50$ runs of the algorithms for each combination of the parameters $k_{max} = \{50, 100, 200, 500, 1000\}$ and $N = \{50, 100, 500\}$. The results of the simulations are summarized in the Tables 2, 3, 4 and 5. In these tables, for each plant and for each combination of $N$ and $k_{max}$, the best $E_{min}(Q)$, the mean and the standard deviation of $E_{min}(Q)$, the success rate (SR) in percentage [%] and the average execution time in seconds [$s$] are shown. The success rate, which is used as a measurement of the algorithm performance, is the ratio of the number of runs with the best solution to the total number of runs of the algorithm $N_{run}$.

The simulation results show that the design methods based on CMA-ES and DE are quite superior than the ones based on FA and PSO. The design method based on CMA-ES gave the best results for both plants, with the smallest performance index value and the highest success rate. The results of the design method based on DE are very good as well. Both methods CMA-ES and DE overcome the design method previously developed based on PSO. And the method based on FA show not to be appropriated for the design of the finite-level dynamic quantizer. In Section 3.7 these results will be discussed in more detail.

On the other hand, the Tables 2 to 5 show that some values, such as the means

Table 2: Simulation results for the second and third order plants by CMA-ES ($N_{run} = 50$ trials). $N$ denotes the number of individuals and $k_{max}$ the maximum number of iterations.

| | | Second order system, $P_1$ | | | | | Third order system, $P_2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | $k_{max}$ | Best | Mean | St. dev. | SR | Time | Best | Mean | St. dev. | SR | Time |
| 50 | 50 | 0.5091 | 0.5870 | 0.1059 | 66 | 5.9 | 0.0719 | 0.0948 | 0.0335 | 0 | 8.8 |
| 50 | 100 | 0.5091 | 0.5395 | 0.0814 | 86 | 12.2 | 0.0691 | 0.0797 | 0.0302 | 82 | 18.2 |
| 50 | 200 | 0.5091 | 0.5800 | 0.1121 | 70 | 18.5 | 0.0691 | 0.0751 | 0.0238 | 94 | 37.8 |
| 50 | 500 | 0.5091 | 0.5438 | 0.0861 | 86 | 19.6 | 0.0691 | 0.0733 | 0.0200 | 92 | 91.6 |
| 50 | 1000 | 0.5091 | 0.5736 | 0.1089 | 74 | 17.5 | 0.0691 | 0.0814 | 0.0328 | 86 | 173.9 |
| 100 | 50 | 0.5091 | 0.5152 | 0.0353 | 98 | 12.7 | 0.0692 | 0.0704 | 0.0009 | 38 | 18.3 |
| 100 | 100 | 0.5091 | 0.5091 | 0.0000 | 100 | 26.7 | 0.0691 | 0.0691 | 0.0000 | 100 | 38.1 |
| 100 | 200 | 0.5091 | 0.5140 | 0.0347 | 98 | 29.8 | 0.0691 | 0.0691 | 0.0000 | 100 | 78.1 |
| 100 | 500 | 0.5091 | 0.5140 | 0.0347 | 98 | 29.0 | 0.0691 | 0.0691 | 0.0000 | 96 | 190.4 |
| 100 | 1000 | 0.5091 | 0.5190 | 0.0486 | 96 | 28.4 | 0.0691 | 0.0691 | 0.0000 | 98 | 388.6 |
| 500 | 50 | 0.5091 | 0.5091 | 0.0000 | 100 | 82.2 | 0.0691 | 0.0691 | 0.0000 | 98 | 86.0 |
| 500 | 100 | 0.5091 | 0.5091 | 0.0000 | 100 | 141.1 | 0.0691 | 0.0691 | 0.0000 | 100 | 180.0 |
| 500 | 200 | 0.5091 | 0.5091 | 0.0000 | 100 | 142.5 | 0.0691 | 0.0691 | 0.0000 | 100 | 356.5 |
| 500 | 500 | 0.5091 | 0.5091 | 0.0000 | 100 | 143.5 | 0.0691 | 0.0691 | 0.0000 | 100 | 911.3 |
| 500 | 1000 | 0.5091 | 0.5091 | 0.0000 | 100 | 146.5 | 0.0691 | 0.0691 | 0.0000 | 100 | 1715.2 |

and even the running time, do not change monotonically. This effect is because the metaheuristics used are stochastic algorithms that will not necessarily give better result by increasing the number of individuals or the number of iterations. In fact, it is an open problem for some metaheuristics to find the optimal values for the number of individuals and the stoping criterias [100].

Next, the performance of the designed quantizer in the system in Figure 6 is evaluated. The quantizer parameters are taken from the simulations performed, in particular, the quantizers that gave the smallest performance indices for each plant are selected. Both quantizers were found for the design method based on CMA-ES, and their values for $P_1$ and $P_2$ are shown as follow:

$$\boldsymbol{\mathcal{A}}_1 = \begin{bmatrix} 0 & 1 \\ -0.7413 & 1.7241 \end{bmatrix}, \quad \boldsymbol{\mathcal{B}}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$
$$\boldsymbol{\mathcal{C}}_1 = \begin{bmatrix} 0.6532 & -1.1619 \end{bmatrix}, \quad d_1 = 3.1082. \tag{77}$$

Table 3: Simulation results for the second and third order plants by DE ($N_{run} = 50$ trials). $N$ denotes the number of individuals and $k_{max}$ the maximum number of iterations.

| | | Second order system, $P_1$ | | | | | Third order system, $P_2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | $k_{max}$ | Best | Mean | St. dev. | SR | Time | Best | Mean | St. dev. | SR | Time |
| 50 | 50 | 0.5091 | 0.6051 | 0.1197 | 62 | 7.8 | 0.0692 | 0.1113 | 0.0461 | 4 | 7.9 |
| 50 | 100 | 0.5091 | 0.5885 | 0.1158 | 68 | 17.2 | 0.0691 | 0.0949 | 0.0419 | 30 | 16.2 |
| 50 | 200 | 0.5091 | 0.5984 | 0.1191 | 64 | 34.7 | 0.0691 | 0.0963 | 0.0433 | 32 | 33.9 |
| 50 | 500 | 0.5091 | 0.5896 | 0.1153 | 68 | 95.1 | 0.0691 | 0.0966 | 0.0432 | 44 | 86.0 |
| 50 | 1000 | 0.5091 | 0.5741 | 0.1086 | 74 | 181.9 | 0.0691 | 0.0838 | 0.0345 | 62 | 175.3 |
| 100 | 50 | 0.5091 | 0.5641 | 0.1026 | 78 | 16.4 | 0.0692 | 0.0871 | 0.0358 | 28 | 16.0 |
| 100 | 100 | 0.5091 | 0.5885 | 0.1158 | 68 | 33.5 | 0.0691 | 0.0817 | 0.0324 | 68 | 33.3 |
| 100 | 200 | 0.5091 | 0.5666 | 0.1023 | 78 | 67.6 | 0.0691 | 0.0853 | 0.0366 | 80 | 69.1 |
| 100 | 500 | 0.5091 | 0.5488 | 0.0910 | 84 | 170.8 | 0.0691 | 0.0832 | 0.0347 | 84 | 175.5 |
| 100 | 1000 | 0.5091 | 0.5438 | 0.0861 | 86 | 349.9 | 0.0691 | 0.0852 | 0.0367 | 84 | 350.9 |
| 500 | 50 | 0.5091 | 0.5190 | 0.0486 | 96 | 69.8 | 0.0692 | 0.0745 | 0.0198 | 76 | 77.5 |
| 500 | 100 | 0.5091 | 0.5438 | 0.0861 | 86 | 139.9 | 0.0691 | 0.0711 | 0.0140 | 98 | 174.6 |
| 500 | 200 | 0.5091 | 0.5425 | 0.0813 | 86 | 286.1 | 0.0691 | 0.0712 | 0.0140 | 96 | 352.5 |
| 500 | 500 | 0.5091 | 0.5438 | 0.0861 | 86 | 686.1 | 0.0691 | 0.0791 | 0.0300 | 90 | 907.0 |
| 500 | 1000 | 0.5091 | 0.5438 | 0.0861 | 86 | 1408.7 | 0.0691 | 0.0691 | 0.0000 | 100 | 1730.8 |

$$\boldsymbol{\mathcal{A}}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.1391 & -1.1584 & 2.0022 \end{bmatrix}, \quad \boldsymbol{\mathcal{B}}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$
$$\boldsymbol{\mathcal{C}}_2 = \begin{bmatrix} -0.2078 & 0.9785 & -1.1550 \end{bmatrix}, \quad d_2 = 3.2337. \tag{78}$$

The selected input signal, which has range $U = [-1, 1]$, is the following:

$$u(k) = 0.7 \sin(3k) + 0.3 \sin(4k). \tag{79}$$

The results are shown in Figure 10, the upper figures show the time responses for the second order system and lower ones for the third order system. Notice that in both cases the quantized input signal $v(k)$ has only two levels since it was specified that $M = 2$. For the second order system Figure 10b shows how the real output $y_q(k)$ follows closely the desired output $y(k)$ and the error between them is very small. In fact, the maximum error registered is $\max_{k\in\{1,2,\ldots,150\}} \text{abs}\,[y_q(k) - y(k)] = 0.3462$. Then, since for the respective designed quantizer $E(Q) = 0.5091$, it is verified that the maximum error in this example is less than the performance

Table 4: Simulation results for the second and third order plants by FA ($N_{run}$ = 50 trials). $N$ denotes the number of individuals and $k_{max}$ the maximum number of iterations.

| | | Second order system, $P_1$ | | | | | Third order system, $P_2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | $k_{max}$ | Best | Mean | St. dev. | SR | Time | Best | Mean | St. dev. | SR | Time |
| 50 | 50 | 0.6469 | 0.9025 | 0.1486 | 0 | 3.2 | 0.1744 | 0.2102 | 0.0233 | 0 | 5.8 |
| 50 | 100 | 0.7562 | 0.7958 | 0.0698 | 0 | 6.4 | 0.1546 | 0.1763 | 0.0148 | 0 | 12.4 |
| 50 | 200 | 0.7322 | 0.7852 | 0.0989 | 0 | 12.5 | 0.1624 | 0.1696 | 0.0031 | 0 | 26.0 |
| 50 | 500 | 0.5320 | 0.7449 | 0.0487 | 4 | 30.6 | 0.1614 | 0.1722 | 0.0078 | 0 | 66.6 |
| 50 | 1000 | 0.5324 | 0.7575 | 0.0922 | 4 | 59.3 | 0.1479 | 0.1716 | 0.0132 | 0 | 134.9 |
| 100 | 50 | 0.7690 | 0.9171 | 0.1569 | 0 | 5.8 | 0.1711 | 0.2063 | 0.0247 | 0 | 12.0 |
| 100 | 100 | 0.7497 | 0.8222 | 0.1311 | 0 | 11.5 | 0.1644 | 0.1735 | 0.0052 | 0 | 26.6 |
| 100 | 200 | 0.6737 | 0.7753 | 0.0587 | 0 | 22.8 | 0.1533 | 0.1699 | 0.0047 | 0 | 55.8 |
| 100 | 500 | 0.5783 | 0.7625 | 0.0442 | 2 | 56.7 | 0.1579 | 0.1692 | 0.0032 | 0 | 144.9 |
| 100 | 1000 | 0.5340 | 0.7629 | 0.0670 | 2 | 113.3 | 0.1438 | 0.1715 | 0.0090 | 0 | 285.7 |
| 500 | 50 | 0.7633 | 0.8128 | 0.0964 | 0 | 28.9 | 0.1668 | 0.1980 | 0.0167 | 0 | 81.2 |
| 500 | 100 | 0.6695 | 0.7821 | 0.0431 | 0 | 56.8 | 0.1479 | 0.1723 | 0.0061 | 0 | 170.2 |
| 500 | 200 | 0.7329 | 0.7620 | 0.0190 | 0 | 113.7 | 0.1341 | 0.1689 | 0.0057 | 0 | 349.3 |
| 500 | 500 | 0.6152 | 0.7672 | 0.0724 | 0 | 282.4 | 0.1200 | 0.1671 | 0.0087 | 0 | 880.0 |
| 500 | 1000 | 0.5684 | 0.7440 | 0.0533 | 8 | 577.9 | 0.1546 | 0.1697 | 0.0037 | 0 | 1776.7 |

index $E(Q)$. The same situation happens for the third order system, where, $\max_{k\in\{1,2,\ldots,150\}} \text{abs}\,[y_q(k) - y(k)] = 0.0319$ and $E(Q) = 0.0691$.

## 3.7 Comparison Among Metaheuristics

In this section the results from the developed simulations are discussed in detail. A comparison among the design methods based on CMA-ES, DE, FA and PSO is developed in terms of the success rate, convergence behavior and execution time. The data for this analysis can be found in the Tables 2, 3, 4 and 5.

### 3.7.1 Success Rate Comparison

For the second order system the best solution found for CMA-ES and DE is the same $E_{min}(Q_{CMA-ES}) = E_{min}(Q_{DE}) = 0.509057$, for PSO is slightly bigger $E_{min}(Q_{PSO}) = 0.509073$ and for FA is quite bigger $E_{min}(Q_{FA}) = 0.531969$. For the third order system this relation is the same $E_{min}(Q_{CMA-ES}) = E_{min}(Q_{DE}) =$

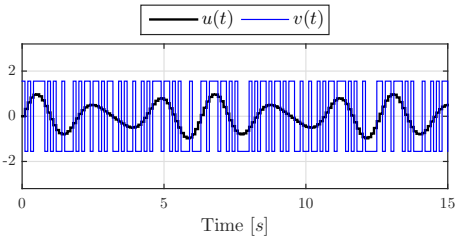
Table 5: Simulation results for the second and third order plants by PSO ($N_{run}$ = 50 trials). $N$ denotes the number of individuals and $k_{max}$ the maximum number of iterations.

| | | Second order system, $P_1$ | | | | | Third order system, $P_2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | $k_{max}$ | Best | Mean | St. dev. | SR | Time | Best | Mean | St. dev. | SR | Time |
| 50 | 50 | 0.7291 | 0.7910 | 0.0271 | 0 | 6.4 | 0.1386 | 0.2052 | 0.0340 | 0 | 5.6 |
| 50 | 100 | 0.6467 | 0.7612 | 0.0196 | 0 | 13.4 | 0.1297 | 0.1814 | 0.0157 | 0 | 11.2 |
| 50 | 200 | 0.7437 | 0.7577 | 0.0027 | 0 | 28.2 | 0.0867 | 0.1650 | 0.0182 | 0 | 23.5 |
| 50 | 500 | 0.7015 | 0.7556 | 0.0080 | 0 | 76.8 | 0.0713 | 0.1579 | 0.0300 | 0 | 66.6 |
| 50 | 1000 | 0.5731 | 0.7518 | 0.0274 | 2 | 159.4 | 0.0706 | 0.1529 | 0.0333 | 0 | 151.7 |
| 100 | 50 | 0.7300 | 0.7815 | 0.0211 | 0 | 10.7 | 0.1047 | 0.1867 | 0.0195 | 0 | 11.4 |
| 100 | 100 | 0.6960 | 0.7593 | 0.0101 | 0 | 22.8 | 0.1556 | 0.1734 | 0.0063 | 0 | 23.7 |
| 100 | 200 | 0.6160 | 0.7491 | 0.0316 | 0 | 49.3 | 0.0789 | 0.1557 | 0.0293 | 0 | 48.5 |
| 100 | 500 | 0.5150 | 0.7476 | 0.0429 | 4 | 140.6 | 0.0716 | 0.1430 | 0.0405 | 0 | 135.7 |
| 100 | 1000 | 0.6261 | 0.7535 | 0.0194 | 0 | 306.0 | 0.0729 | 0.1510 | 0.0362 | 0 | 299.5 |
| 500 | 50 | 0.5204 | 0.7550 | 0.0366 | 2 | 52.8 | 0.0920 | 0.1660 | 0.0209 | 0 | 56.6 |
| 500 | 100 | 0.5561 | 0.7447 | 0.0436 | 4 | 110.7 | 0.0774 | 0.1505 | 0.0342 | 0 | 115.0 |
| 500 | 200 | 0.5114 | 0.7458 | 0.0477 | 4 | 234.5 | 0.0712 | 0.1491 | 0.0374 | 0 | 249.2 |
| 500 | 500 | 0.5097 | 0.7025 | 0.0964 | 22 | 566.6 | 0.0709 | 0.1357 | 0.0426 | 0 | 698.6 |
| 500 | 1000 | 0.5091 | 0.6777 | 0.1132 | 32 | 1108.1 | 0.0701 | 0.1305 | 0.0453 | 0 | 1477.3 |

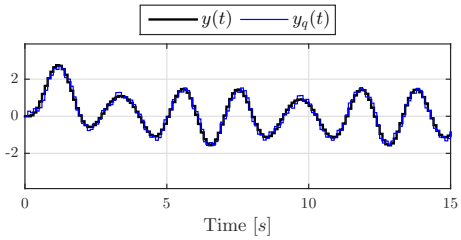
0.069132, $E_{min}(Q_{PSO}) = 0.070130$ and $E_{min}(Q_{FA}) = 0.119999$.

The best solutions are provided by CMA-ES and DE. These solutions are the same in both cases indicating that they both have the exploration capacities to solve the quantizer design problem. Now, in the case of PSO for the second order system, the best solution is very close to the one obtained by CMA-ES and DE, while for the third order system, the difference between the results is not negligible. In contrast, the solutions provided by FA are relatively big in comparison with the others.

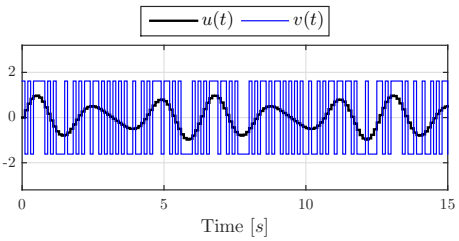
A more important point is the success rate when comparing these algorithms. The success rate gives an idea about how effective is an algorithm. A low success rate indicates that the algorithm gets trapped into local minima, and that the probability to find the global minimum is small. Thus, an algorithm with high success rate is reliable. The success rates of the quantizer design algorithms are shown in Figure 11 for the second order system and in Figure 12 for the third order system, this values are the same as in the tables.



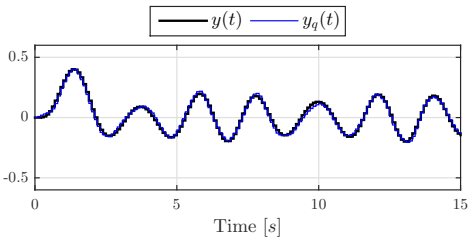

(a) Input signals of 2nd order plant $P_1$.




(b) Output signals of 2nd order plant $P_1$.




(c) Input signals of 3rd order plant $P_2$.




(d) Output signals of 3rd order plant $P_2$.

Figure 10: System responses when the control signal $u(k)$ is applied to the plants $P_1$ and $P_2$. The black lines represent the signals of the system without quantization. Meanwhile, the blues ones are the signals when the quantization is performed.

They show how the performances of CMA-ES and DE are quite better than the performances of PSO and FA for the design of the quantizer. For the second order system the success rates of CMA-ES and DE are always over 60%. Meanwhile, the success rates of PSO are less than 40% and for FA less than 10%. For the third order system the success rates of PSO and FA are 0% in all the cases. Thus, it is verified that the quantizer design based on CMA-ES and DE are reliable methods while the ones based on PSO and FA are not. Now, for both plants the success rates obtained with CMA-ES are better than the ones obtained with DE, specially when the number of individuals $N$ and (or) the maximum number of generations $k_{max}$ are small, for bigger values of $N$ and $k_{max}$ their success rates are very close to each other. Then, it is fair to say that the design method based on CMA-ES is better and more reliable than the one based on DE.

An interesting result from the tables is the values of the mean. The means were

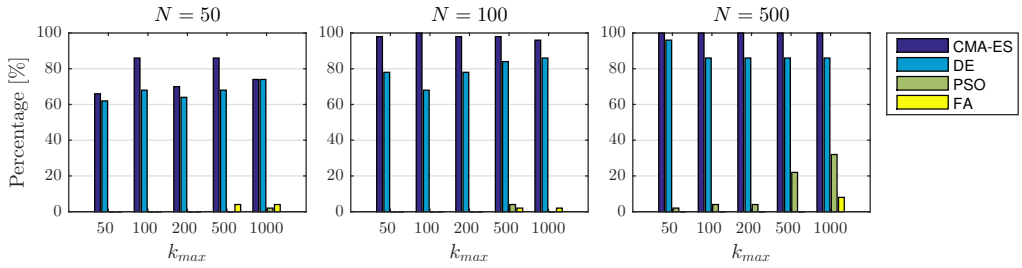



Figure 11: Success rate for the second order plant ($P_1$).

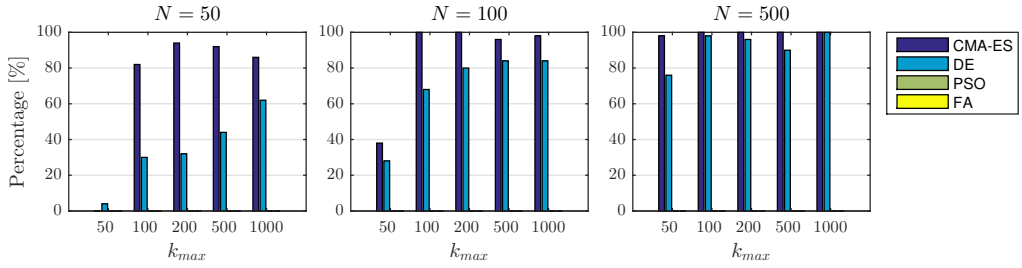



Figure 12: Success rate for the third order plant ($P_2$).

taken over the $N_{run} = 50$ trials of the algorithm. For the CMA-ES based method the mean for both order systems is very close to the best value or $E_{min}(Q)$, same for DE, but not as close as in CMA-ES. In contrast, for PSO the best value and the mean are different for the second order system, since the mean is closer to the local minima, in the third order case they are both close to the local minima. The behavior of FA is similar to PSO. Another remarkable fact taken from the tables is that the standard deviation in the CMA-ES based method is very small, almost zero, indicating how close the results are from the mean that at the same time is almost as the best result. The success rates are a reflection of these results.

### 3.7.2 Convergence Time Comparison

The convergence behavior of the algorithms is shown in Figure 13 for the second order system and in Figure 14 for the third order system by sequences of $E(Q)$ against the generation number $k$. In these figures there were selected examples of trials of the algorithms for each value of $N$ and $k_{max} = 500$, although in the figures

the sequences are shown until $k = 50$. The only criteria used for the selection of this sequences among the respective 50 trials was that the initial $E(Q)$ should be close to $E(Q) = 1$. In each case the figures show that the methods based on CMA-ES and DE converge fast to the global optima, and that the design methods based on PSO and FA converge more slowly to the local optima.

In terms of execution time PSO and FA shows to be faster than DE for both systems, considering the results of Table 3, 4 and 5. However, comparing PSO and FA execution times, it is notable that for the second order system the FA algorithm is quite faster than PSO but for the third order system FA is slower than PSO. This is due to the time complexity of the algorithms, thus, increasing the dimension of the problem will make FA even more slower than PSO. For CMA-ES the situation is a little ambiguous. For the second order system, it shows to be quite faster than the others methods, moreover the execution time does not depend on $k_{max}$ after a certain value. On the other hand, for the third order system, CMA-ES is slower than the other algorithms, and the execution time increases with $k_{max}$. This behavior is explained with the double termination condition of the CMA-ES algorithm that are implemented in this study. The first one is that the generation number $k$ reaches $k_{max}$, this is the only termination condition for DE, FA and PSO, and this explains why for the third order system the execution time depends on $k_{max}$. The second termination condition of CMA-ES is that the condition number of the covariance matrix $\mathbf{\Sigma}$ exceeds $10^{14}$, clearly this condition is responsible for the behavior in the case of the second order system. The CMA-ES policy is that the algorithm should be stopped whenever it becomes a waste of CPU-time to continue. Then, it is possible to add more termination conditions for the CMA-ES algorithm, and in this way reduce its execution time.

## 3.8 Summary

In this chapter design methods for the finite-level dynamic quantizer subjected to data rate constraints are proposed. These methods are based on CMA-ES, DE and FA. By means of numerical simulations the effectiveness of the proposed design methods were confirmed. The metaheuristics that showed the best performance in terms of success rate and execution time are the CMA-ES and DE

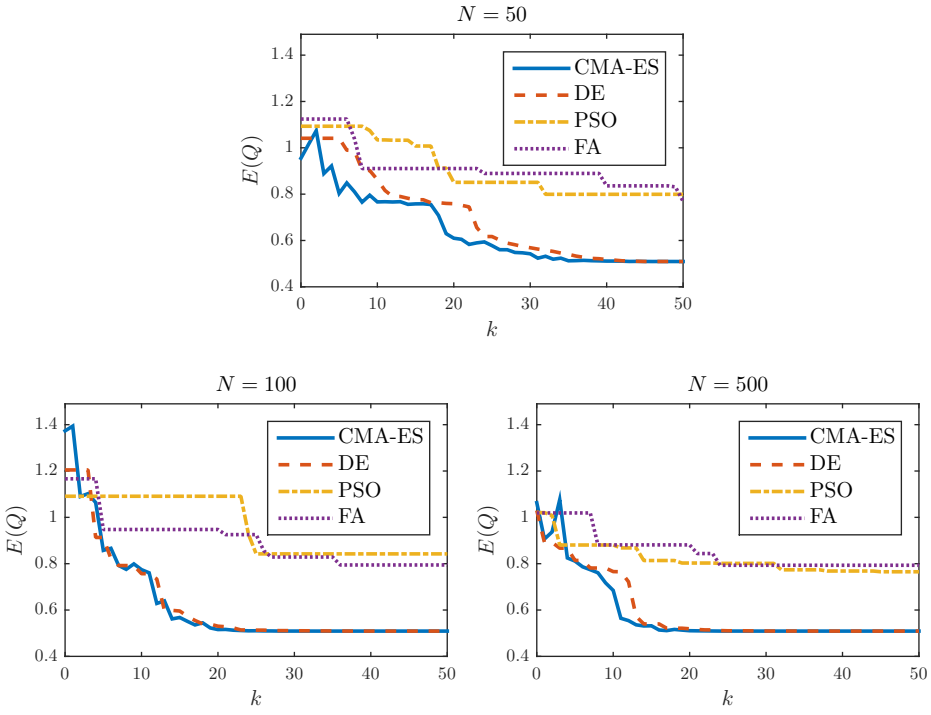



Figure 13: Convergence behavior of the different metaheuristics for the second order plant $P_1$ ($k_{max} = 500$). For better visualization the graphics only show the results until $k = 50$. $E(Q)$ further decreases in some cases.

algorithms.

The CMA-ES algorithm is not easy to implement, but it carries the big advantage that it does not require the tuning of any control parameter. Their values are already optimized and established by default. DE is very easy to implement, it is fast and has only three control parameters. These methods show very good performances even with a relatively small amount of individuals and number of generations. For CMA-ES the execution time shows a bipolar behavior for the different order systems, although this can be solved by adding more termination conditions to the algorithm.

Compared to the other metaheuristic based design methods, it was verified that the performances of the CMA-ES and DE based methods are quite better than the ones based on PSO and FA. However, compared with each other the method based on CMA-ES is better than the one based on DE. Although,

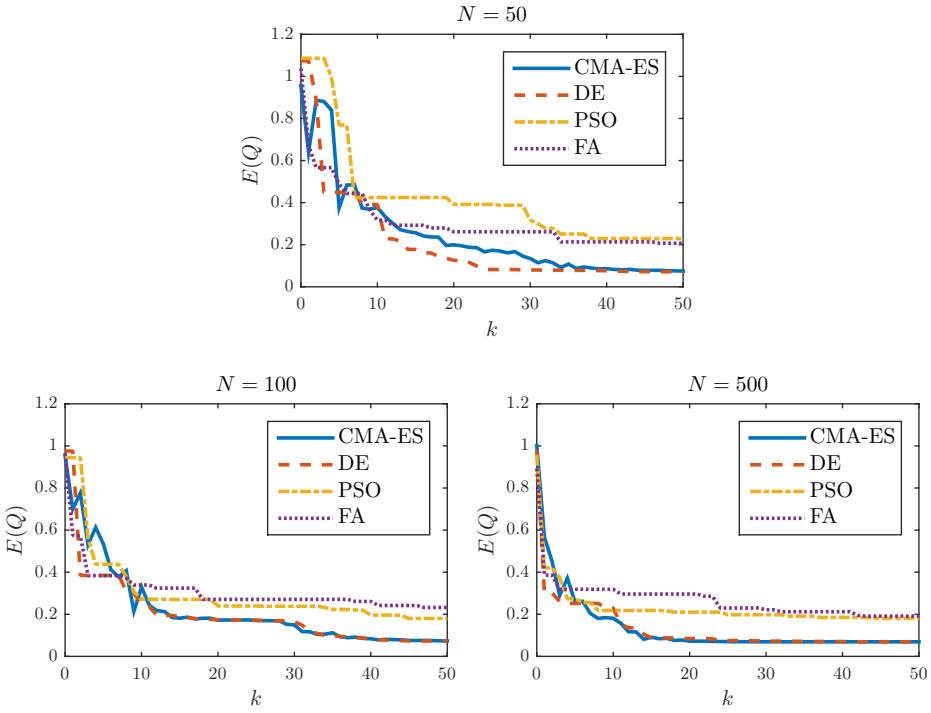



Figure 14: Convergence behavior of the different metaheuristics for the third order plant $P_2$ ($k_{max} = 500$). For better visualization the graphics only show the results until $k = 50$. $E(Q)$ further decreases in some cases.

they show similar success rates and convergence behavior there are some small differences.

# 4. Design of Event-triggered Quantizers (ETQs)

In this chapter, a switching-type finite level dynamic quantizer aimed to the reduction of the NCSs' network traffic is introduced. In addition, some design methods of this quantizer based on differential evolution are proposed. With numerical simulations it is verified that the proposed quantizer and its design methods are effective. Finally, the design methods are compared among each other and several conclusions drawn.

## 4.1 Introduction

In conventional control systems the controllers send signals to the plant in each sampling time, which leads to continuous traffic in the network. Bandwidth limitations and energy constraints set severe restrictions on the operation of many types of NCSs. In addition, network traffic congestion brings several problems such as delays in the delivery of data and packages loss. Then, in order to reduce the effects of this problems, one admissible approach is to design a quantizer that sends data to the plant only when it is necessary. This approach is similar to the ones developed in the event-triggered control field where an event generator attached to the plant asks for data to the controller when some kind of variation in the plant's state or output happens [44, 69, 95].

There are many studies that consider the reduction of network traffic generated by the control system. The event-triggered and self-triggered control disciplines are being developed with that goal. Nevertheless, there are not so many studies that consider this traffic reduction in quantized NCSs. Some studies worth to be mentioned are summarized as follow. [120] considered the problem of event-triggered $H_\infty$ control for a quantized singular NCS. [99] developed a triggering method combining the event-triggering bandwidth reduction strategies and speech coding techniques. [30] combined model-based NCSs and event-triggered control to stabilize dynamical systems subjected to quantization and network delays. [51] proposed a delay system model where the criteria of stability and control of event-triggered NCSs are established using linear matrix inequalities (LMIs). In addition, [79] tackled the considered problem without using event-triggering techniques. Instead this study uses model-based NCSs and derives

sufficient stability conditions for two types of static and a dynamic quantization schemes.

The present study is inspired in the event-triggered techniques mentioned above. To tackle the problem of network traffic reduction, this chapter proposes a class of switching-type dynamic quantizer denominated *event-triggered quantizer* (ETQ). This quantizer is composed of a feedback type dynamic quantizer and a Gaussian function based switching mechanism. Then, this study finds a quantizer that (i) minimizes the deterioration of the system's performance due to quantization, (ii) satisfies the constraints caused by the communication data rate, and (iii) reduces the amount of data that the system puts in the network. The design of this quantizer is carried out using the differential evolution metaheuristic algorithm due to the complexity of its design problem.

Differently from the previous studies, this chapter does not concern with the stability or the control of the plant, nor with the design of the controller. An open loop system with a stable plant is considered. Then, the focus of this study is on the minimization of the performance degradation due to quantization and the reduction of the network traffic. An advantage of the proposed quantizer is that the designer can specify the desired amount of network traffic reduction by setting a parameter called network utilization rate during the quantizer's design. Then, the quantizer will balance the amount of traffic that is saved in the network with the system's performance degradation level. Another advantage is that this study takes into account the data rate constraints of the channel, a feature disregarded in the previous studies.

The contributions of this chapter are as follows. First, an event-triggered quantizer with a Gaussian function based trigger mechanism is presented. Although the structure with Gaussian functions has the potential to achieve satisfactory performance due to its flexibility, it has not been studied so far to the best of the author's knowledge. Then, a network utilization rate is introduced, and an optimal quantizer design problem for network traffic reduction is formulated. Thus, a solution to the design problem could reduce the network traffic, which leads to the improvement of the network reliability. The obtained results facilitate the design of practical dynamic quantizers for NCSs.

The content of this chapter was published in [108], [109] and [112].

## 4.2 Problem Formulation

### 4.2.1 System Description

This chapter considers the system shown in Figure 15. This system is composed of the event-triggered quantizer $Q_S$, the plant $P$, and the communication channel. It is assumed that the channel has no losses or delays, that the plant is a remote control system with an inner loop, and that the reference signal $u$ is bounded, i.e., $u(k) \in U$ for $U := [u_{min}, u_{max}]$.

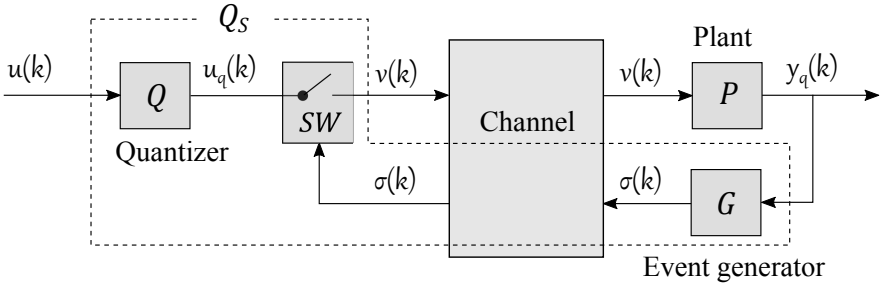



Figure 15: NCS with an event-triggered quantizer.

The plant $P$ is linear, discrete-time, and sigle-input-single-output (SISO). It is same as the one described in Section 3.1 represented by Equation (1). In this case it is also assumed that the plant is stable. Note that Equation (1) can express a closed loop system with a plant and a stabilizing controller.

The structure of $Q_S$ is shown in Figure 15. It is composed of a feedback-type dynamic quantizer $Q$, an event generator $G$, and a switching mechanism $SW$. The quantizer $Q$ is the one described by Equation (2) in Section 3.1 with the difference that its output is represented by $u_q(k)$ instead of $v(k)$. To avoid confusion, the equation with the modified notation is depicted as follows

$$Q : \begin{cases} \boldsymbol{\xi}(k+1) & = \boldsymbol{\mathcal{A}}\boldsymbol{\xi}(k) + \boldsymbol{\mathcal{B}}(u_q(k) - u(k)), \\ u_q(k) & = \mathrm{q}[\boldsymbol{\mathcal{C}}\boldsymbol{\xi}(k) + u(k)]. \end{cases} \tag{80}$$

The considerations are the same as the one in Equation (2), including the structure of the static quantizer q[·] exemplified in Figure 7.

The event generator $G$ produces a trigger signal $\sigma(k) \in \{0, 1\}$ that operates as follows. When an event occurs, $G$ sends $\sigma(k) = 1$ to $SW$ on the other side of the channel and the switch closes allowing $Q$ to send $v(k) = u_q(k)$ to the plant.

Notice that the switch remains normally open, and that $\sigma(k) = 0$ means that no signal is sent from $G$. If no data reach the plant in a sampling time $k$, the plant will hold its previous input, namely, $v(k) = v(k-1)$. This behavior is illustrated in Figure 16.

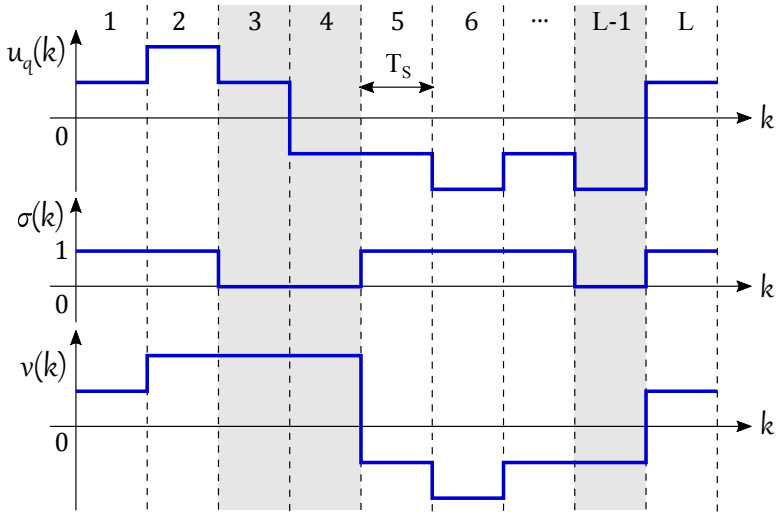



Figure 16: Operation of the trigger signal $\sigma(k)$.

Accordingly, the operation of $SW$ can be represented as

$$SW : \begin{cases} \zeta(k+1) & = v(k), \\ v(k) & = \sigma(k)u_q(k) + (1-\sigma(k))\zeta(k), \end{cases} \tag{81}$$

where, $\zeta \in \mathbb{R}$ is the state of $SW$ with $\zeta(0) = 0$, and $v(k)$ is the output of $Q_S$.

The events in $G$ are determined by the output signal $y_q(k)$ using the following rule

$$G : \qquad \sigma(k) = \begin{cases} 1 & \text{if} \quad H[y_q(k)] \geq 0, \\ 0 & \text{otherwise}, \end{cases} \tag{82}$$

where $H[y_q(k)]$ is the *trigger function* defined as

$$H[y_q(k)] := \boldsymbol{w}^\top \boldsymbol{\phi}[y_q(k)] = \sum_{i=1}^{n_G} w_i \exp\left[-\frac{(y_q(k)-\mu_i)^2}{2s_i^2}\right], \tag{83}$$

where $\boldsymbol{\phi}(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_{n_G}(x)]^\top$ is a vector of Gaussian functions, and $\boldsymbol{w} \in \mathbb{R}^{n_G}$ is a vector of constant weights. There are in total $n_G$ Gaussian functions

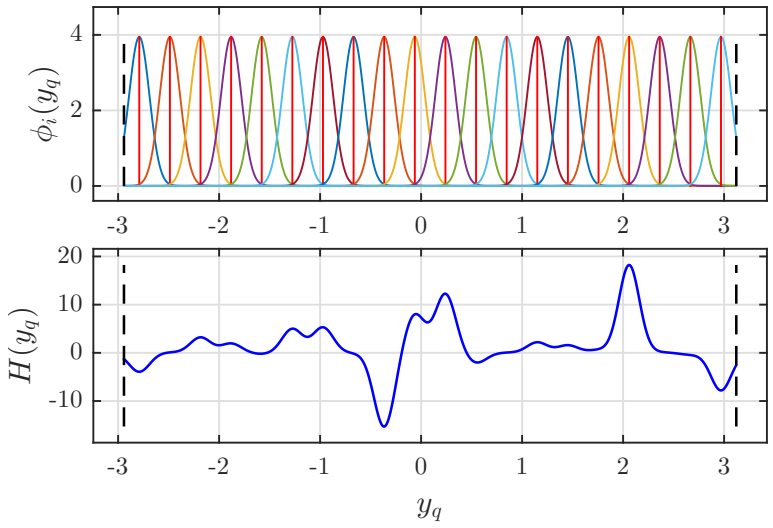



Figure 17: Example of a trigger function $H(y_q)$ and the Gaussian functions $\phi_i(y_q)$ that compose it for $n_G = 20$. The vertical brake lines represent $y_{min}$ and $y_{max}$.

distributed along the range of the output signal $y(k) \in [y_{max}, y_{min}]$ and together they compose the trigger function as Figure 17 shows. Each Gaussian function has only two parameters, a mean $\mu_i$ and a variance $s_i^2$ that are gathered in the vectors $\boldsymbol{\mu}$ and $\boldsymbol{s}^2$, respectively. This type of structure for the trigger function is used due to its simplicity, flexibility, and relatively small number of parameters.

In this study it is assumed that only $M$ is given. Thus, the design parameters for $Q_S$ are $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, $\boldsymbol{\mathcal{C}}$, $d$, $\boldsymbol{\mu}$, $\boldsymbol{s}^2$, and $\boldsymbol{w}$.

Notice that there are other versions of ETQ that were considered. These versions are described and commented in Section 4.4. The version that was explained in this section is the one that exhibits the best performance.

### 4.2.2 Performance Index

The quantizer's performance analysis is carried out by using the *error system* shown in Figure 18. The error system's lower branch represents the ideal case in which $u(k)$ is applied directly to the plant giving the *ideal output* $y(k)$. In the upper branch the effects of the channel and event-triggered quantization are considered. The quantized reference signal $v(k)$ is applied to the plant giving the

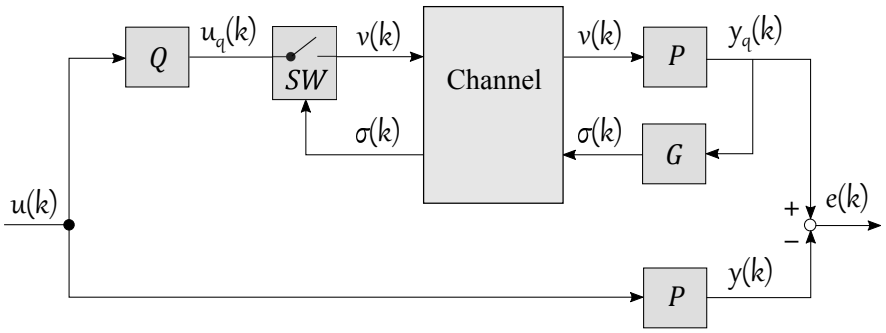



Figure 18: Event-triggered quantizer's error system.

output $y_q(k)$. The difference between these outputs is the error signal $e(k) = y_q(k) - y(k)$.

The biggest possible value of $e(k)$, i.e., the worst case performance of the system, is known as *performance index* $E(Q_S)$ which is defined as in Section 2.1.1 by Equation (3). In order to obtain the smallest performance degradation, it is necessary to make $E(Q_S)$ as small as possible by the appropriate design of the event-triggered quantizer $Q_S$. Then, the dynamic quantizer design is reduced to an optimization problem, where the objective function is given by $E(Q_S)$ in (3) and the variables are the design parameters of $Q_S$.

However, because the quantizer's structure is different the result given by Equation (4) to calculate $E(Q_S)$ cannot be used. Instead, this study evaluates the finite time performance and considers an specific input signal. Thus, the following approximated version of $E(Q_S)$, with a finite evaluation interval $L$ and for a given $u(k)$, is used

$$\tilde{E}(Q_S) := \max_{k\in\{1,2,\ldots,L\}} \operatorname{abs}\left(y_q(k) - y(k)\right). \tag{84}$$

Notice that the relation $E(Q_S) \geq \tilde{E}(Q_S)$ holds since $E(Q_S)$ represents the maximum possible error in the output at any time and for any signal $u(k)$. Additionally, notice that $\tilde{E}(Q_S)$ depends not only on $Q_S$ but also on $u$ and $\boldsymbol{x}_0$. Although the notation $\tilde{E}(Q_s, u, \boldsymbol{x}_0)$ should be used, $u$ and $\boldsymbol{x}_0$ are omitted for simplicity.

### 4.2.3 Data Rate Constraint

In this chapter it is considered that the channel is subjected to data rate constraints, as it was described in Section 3.3. This limitation affects the design of quantizers by subjecting them to the following condition $M \leq 2^{N_b}$. It is also assumed that $M$ is given satisfying this data rate constraint.

If there is no saturation in a finite-level static quantizer, then the maximum quantization error is given by $\Delta q = d/2$. In order to minimize $\tilde{E}(Q_S)$ it is necessary to make $\Delta q$ as small as possible. As showed in Section 3.3, the smallest $d$ that satisfies the data rate constraints and does not saturate the static quantizer is the one given by Equation (8), where $\bar{\Lambda}$ is given by Equation (31) and $\mathbf{T}$ is the matrix composed of the right eigenvectors of $\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{BC}}$ as columns that is used for diagonalization [90].

When the switch $SW$ closes the feedforward system is the same as the one considered in Chapter 2, for this reason the result in Equation (8) can be used in the design of $Q_S$. Thus, in this study $d^*$ is used to construct the event-triggered quantizer.

### 4.2.4 Network Utilization Rate

Normally the quantizer operates periodically, sending data through the network in every sampling time. In contrast, the event-triggered quantizer sends data only when the event generator asks for it. In this study the amount of traffic that the quantizer puts in the network is represented in percentage and it is called *network utilization rate* or NUR. Considering that $L$ is the total number of sampling intervals, as Fig. 16 shows, the NUR can be defined as

$$\Psi := \frac{100}{L} \sum_{k=1}^{L} \sigma(k). \tag{85}$$

For instance, in Figure 16 if $L = 9$, then $\Psi = 66.67\%$.

This study aims to reduce the amount of network traffic by the appropriate design of $Q_S$. Of course, the reduction of the traffic in the network will negatively affect the performance of the system by increasing the amplitude of $e(k)$. Then, it is important to make a compromise between the amount of traffic put in the network and the precision of the output.

The traffic reduction is performed by establishing the desired *maximum* NUR, reprented by $\Psi^*$, as a design parameter for the event-triggered quantizer. Thus, the $\Psi^*$ is given by the designer.

### 4.2.5 Quantizer Design Problem

The design of $Q_S$ is formulated as an optimization problem where $\tilde{E}(Q_S)$ is minimized under certain conditions. Thus, the design problem is formalized as follows:

Suppose that $P$, $M$, $\Psi^*$, $u(k)$, and $U$ are given. Then, find the parameters of $Q_S$: $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, $\boldsymbol{\mathcal{C}}$, $d$, $\boldsymbol{\mu}$, $\boldsymbol{s}^2$, and $\boldsymbol{w}$ which minimize $\tilde{E}(Q_S)$, under the conditions that:

i The dynamic quantizer $Q$ is stable ($\bar{\Lambda} < 1$),

ii The data rate constraint is satisfied ($d = d^* > 0$),

iii The system's NUR is bounded ($\Psi \leq \Psi^*$),

iv The variances are positive ($s^2_{min} > 0$), and

v The means are in the range of the output signal ($\mu_{max} \leq y_{max}$ and $\mu_{min} \geqslant y_{min}$,).

The $Q_S$ resulting from this problem, besides minimizing the output error, is stable, satisfies the communication rate constraint and limits the amount of traffic generated by the system.

This design problem cannot be solved by conventional optimization methods due to its nonlinear and nonconvex nature. Hence alternative methods such as metaheuristics should be used. In Chapter 3 the differential evolution metaheuristic algorithm (DE) has been employed successfully showing a very good performance in the design of dynamic quantizers. Moreover, DE is implemented easily, is fast and has few control parameters. For these reasons, in this chapter, DE is adopted to solve the event-triggered quantizer's design problem. This study uses the classical DE/best/1/bin strategy for the optimization.

Notice that in this chapter DE is preferred over CMA-ES for the design of ETQs. The reason for this is that, although CMA-ES showed better performance than DE in the previous chapter, CMA-ES consumes more computational resources than DE. This is specially important when optimizing high dimensional

problems. In the previous chapter the dimension of the problems were $n = 4$ and $n = 6$ for the second order and third order plants respectively. However, in this chapter the dimension of the optimization problem goes until $n = 64$. Thus CMA-ES becomes slower and it is more difficult to run many simulations at the same time in computers with moderate performance. On the other hand, DE does not suffer as much a CMA-ES in this situation and it still runs relatively fast.

The proposed design problem of $Q_S$ is subjected to several constraints and DE has no direct way to handle them. Therefore, the method developed by [73] explained in Section 2.2 is used to transform the constrained optimization problem into an unconstrained one. The unconstrained problem is the following

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}}\, J(\boldsymbol{\theta}), \qquad \text{for} \qquad J(\boldsymbol{\theta}) := \begin{cases} J_v(\boldsymbol{\theta}) & \text{if } p(\boldsymbol{\theta}) < 0, \\ p(\boldsymbol{\theta}) & \text{otherwise,} \end{cases} \tag{86}$$

$$J_v(\boldsymbol{\theta}) := \arctan[\tilde{E}] - \pi/2 \quad \text{with} \quad d = d^*, \tag{87}$$

$$p(\boldsymbol{\theta}) := \max[(\bar{\Lambda} - 1),\ -d^*,\ (\Psi - \Psi^*),\ -s^2_{min}, (\mu_{max} - y_{max}),\ (y_{min} - \mu_{min})], \tag{88}$$

where $\tilde{E}$, $d^*$, $\bar{\Lambda}$, and $\Psi$ correspond to the Equations (84), (8), (31), and (85), respectively. In addition, $s^2_{min}$, $\mu_{min}$, $\mu_{max}$, $y_{min}$, and $y_{max}$ are the minimum and maximum values of $\boldsymbol{s}^2$, $\boldsymbol{\mu}$ and the range of $y(k)$. The design problem constraints are codified in (88). Notice that the design variable $\boldsymbol{\theta} \in \mathbb{R}^n$ is constructed with the $n$ unknown elements of the matrices $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, $\boldsymbol{\mathcal{C}}$, $\boldsymbol{\mu}$, $\boldsymbol{s}^2$, and $\boldsymbol{w}$.

### 4.2.6 Event Definition

Notice that despite the fact that the structure and design method of the event-trigered quantizer $Q_S$ was described in this section there is still no definition of an *event*. This study does not provide such definition. An event happens when the triggered function $H[y_q(k)]$ defined in Equation (83) becomes bigger or equal than zero. This function is built by adding together the weighted Gaussian functions. The weights, means and variances of these Gaussian functions are determined by the metaheuristic algorithm. Thus, in a sense, is the metaheuristic algorithm the one that determines what an event is.

There are several studies in the event-triggered and self triggered control fields that provided different definitions of events for different types of systems [44, 69, 95]. However, this study does not use those definitions, since a new type of quantizer is proposed. It is left as future work to consider formal definitions of events and to analyze the meaning from the NCSs perspective of the events that are determined from the metaheuristic algorithms.

## 4.3 Numerical Examples

Several numerical examples were carried out to prove that the event-triggered quantizer and the proposed design method show good performance. The plant used in these examples is given by

$$P(s) \;=\; \frac{s+20}{s^2+3s+2}. \tag{89}$$

This plant is discretized using a zero-order hold and a sampling time $T_S = 0.1[s]$. It is the same as the one in Equation (73) used in Section 3.6. The given parameters of $Q_S$ are: $M = \{2,\ 4,\ 8,\ 16\}$, $L = 200$, $n_G = 20$, $U = [-1, 1]$ and $\Psi^* = \{100\%,\ 90\%,\ 70\%\}$. The quantizer is designed for the following signal

$$u(k) = 0.3\sin(6k) + 0.4\sin(k) + 0.3\sin(3k). \tag{90}$$

The target vectors used in the DE algorithm are built with the design parameters of $Q_S$. They are arranged as follows

$$\begin{aligned}
\boldsymbol{\mathcal{A}} &= \begin{bmatrix} 0 & 1 \\ \theta_1 & \theta_2 \end{bmatrix}, \quad \boldsymbol{\mathcal{B}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{\mathcal{C}} = \begin{bmatrix} \theta_3 & \theta_4 \end{bmatrix}, \\
\boldsymbol{\mu} &= \begin{bmatrix} \theta_5 & \theta_6 & \dots \theta_{24} \end{bmatrix}^\top, \quad \boldsymbol{s}^2 = \begin{bmatrix} \theta_{25} & \theta_{26} & \dots & \theta_{44} \end{bmatrix}^\top, \\
\boldsymbol{w} &= \begin{bmatrix} \theta_{45} & \theta_{46} & \dots & \theta_{64} \end{bmatrix}^\top.
\end{aligned} \tag{91}$$

Thus, a target vector is given by $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \dots\ \theta_{64}]^\top$ and the dimension of the optimization problem is $n = 64$. The control parameters of DE are $N = 500$, $k_{max} = 2000$, $F = 0.6$, and $H = 0.9$. In addition, the individuals are randomly initialized within the range $S_0 = [-1, 1]^n$ and the simulations were performed $N_{run} = 10$ times for each considered case.

From these simulations several implementations of $Q_S$ were obtained. Each of them is optimized for a combination of the parameters $M$ and $\Psi^*$. To show that the designed quantizers work properly the signal in Equation (90) is applied to the error system in Figure 18. For example, Figure 19 shows the signals resulting from the quantizer designed with $M = 4$ and $\Psi^* = 70\%$. It can be seen that the output signals $y(k)$ and $y_q(k)$ are very similar, despite the fact that the channel is used only 70% of the time, as $\sigma(k)$ shows. In general, the designed quantizers show good performance. Furthermore, the channel's data rate constraint is satisfied. This is evidenced by the fact that signal $u(k)$ is inside the range of $v(k)$, which indicates that no saturation occurs in $Q$.

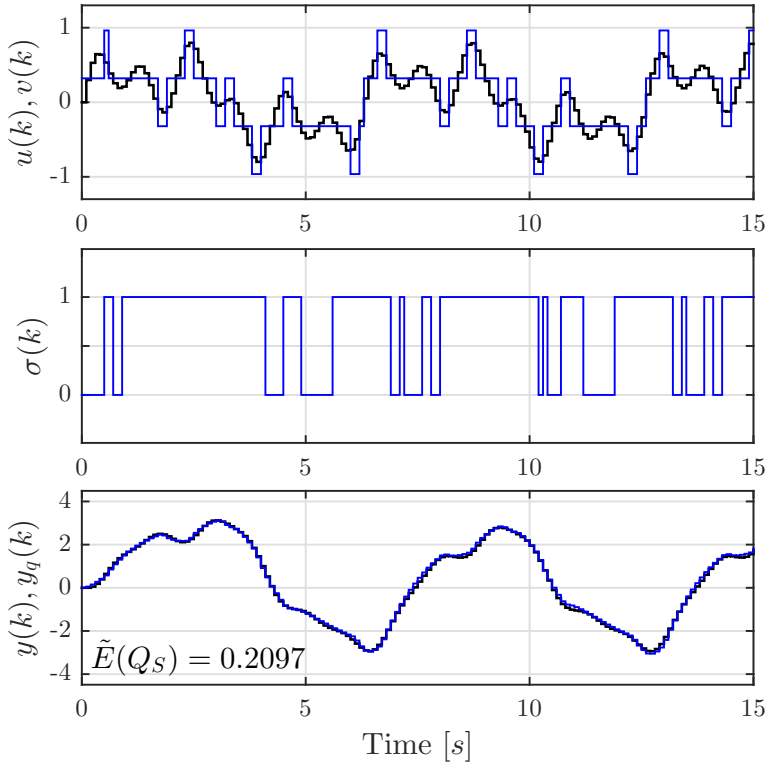



Figure 19: Signals obtained when $u(k)$ is applied to the system with $Q_S$ designed for $M = 4$ and $\Psi^* = 70\%$. The black lines are the ideal signals $u(k)$ and $y(k)$, and the blue ones are the signals obtained when the event-triggered quantization is performed $v(k)$, $\sigma(k)$ and $y_q(k)$.

Figure 20 offers another perspective to further validate the effectiveness of $Q_S$. This figure shows the error signals $e(k)$ obtained by applying random trigger signals $\sigma(k)$ to the optimal quantizer $Q$ designed to minimize the performance index but with no regard of the traffic generated. In these examples the optimal quantizer is designed for $M = 4$ and the reference signal is the one in Equation (90). For comparison purposes the error signals obtained for a random $\sigma(k)$ with NUR close to 70% are selected. These NURs are $\Psi_1 = 69\%$, $\Psi_2 = 71\%$ and $\Psi_3 = 67\%$ for $e_1(k)$, $e_2(k)$ and $e_3(k)$, respectively. Then, the error signal $e_{70\%}(k)$ obtained with the $Q_S$ designed for $M = 4$ and $\Psi^* = 70\%$ is included in the graphic. Clearly, $e_{70\%}(k)$ is the smallest error signal meaning that the event-triggered quantizer effectively reduces the system's performance degradation and the system's NUR at the same time.

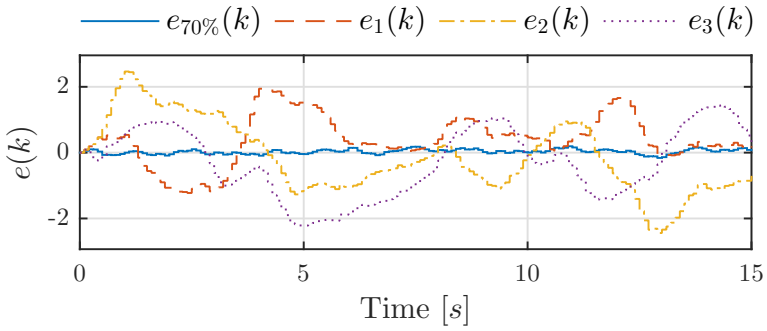



Figure 20: Error signals $e_i(k)$ $(i = 1, 2, 3)$ obtained by applying random trigger signals $\sigma_i(k)$ $(i = 1, 2, 3)$ to the optimal quantizer $Q$ designed for $M = 4$. The error signal $e_{70\%}(k)$ is the one obtained using the $Q_S$ designed for $M = 4$ and $\Psi^* = 70\%$.

Figure 21 depicts a comparison among the error signals $e(k)$ obtained with the quantizers designed for different $\Psi^*$s. The error signal's magnitude decreases when $M$ increases as can be verified by comparing the graphics for $M = 4$ and $M = 8$. Also, it can be verified how the error signals increase their magnitude as $\Psi^*$ decreases. In addition, notice that $\Psi^* = 100\%$ is the optimal case in which no event-triggering is performed and it gives the smallest error signal that can be obtained using $Q_S$. Then, the differences among the error signal magnitudes for the cases with $\Psi^* = 90\%$, $\Psi^* = 70\%$ and $\Psi^* = 100\%$ are not very big. This means

that the performance of $Q_S$ is comparable to the case with no event-triggering, specially for high values of $\Psi^*$.

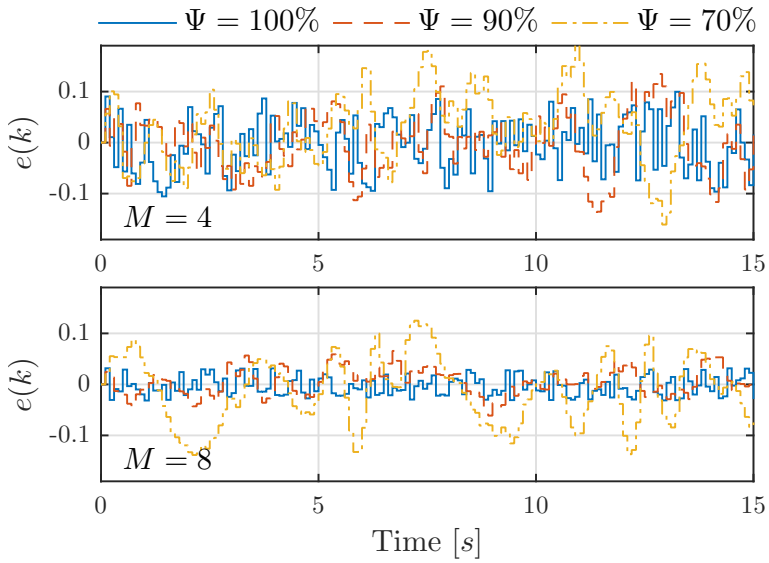



Figure 21: Error signals $e(k)$ when $u(k)$ is applied to the error system with the $Q_S$ designed for different values of $\Psi^*$ and $M$.

## 4.4 Comparison among different versions of ETQ

In this section a comparison among the proposed design method (M1) and three other design methods is carried out. The methods are summarized in Table 6.

The methods M2 and M3 were considered in [108], meanwhile M1 and M4 were published in [112]. In Table 6, the methods are classified into 1 Step and 2 Steps methods. In the 1 Step methods the dynamic quantizer $Q$ is designed simultaneously with the event-generator $G$ using DE. In the 2 Steps methods, first, the optimal quantizer $Q = Q^*$ that minimizes $E(Q)$, Equation (3), and satisfies the data rate constraint is obtained using the method developed in Chapters 2 and 3. Then, in the second step, $G$ is calculated using DE for $Q^*$ and having in account that $\Psi \leq \Psi^*$.

Moreover, the ETQs are divided into VV and V0 that represent two types of operation of the quantizer. In the VV type, the plant assumes $v(k) = v(k-1)$

Table 6: Differences among the design methods of $Q_S$.

| Method | M1 | M2 | M3 | M4 |
|---|---|---|---|---|
| Type | 1 Step<br>VV | 1 Step<br>V0 | 2 Steps<br>V0 | 2 Steps<br>VV |
| Given | $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, M, \Psi^*, u(k), U$ | | | |
| | | $\boldsymbol{\mu},\ \boldsymbol{s}^2$ | $\boldsymbol{\mu},\ \boldsymbol{s}^2$ | |
| Find | $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, d, \boldsymbol{w}$ | | | |
| | $\boldsymbol{\mu}, \boldsymbol{s}^2$ | | | $\boldsymbol{\mu}, \boldsymbol{s}^2$ |

when it does not receive an input in a sampling time, i.e., $\sigma(k) = 0$. This is the case represented by $SW$ in Equation (81). In the V0 type, the plant assumes $v(k) = 0$ when $\sigma(k) = 0$. In this case, $v(k)$ can be presented simply as $v(k) = \sigma(k)u_q(k)$. An example of the V0 type of operation is illustrated in Figure 22. In order to clarify the difference between the V0 and VV types of ETQs, the signals $u_q(k)$ and $\sigma(k)$ in Figure 22 are the same as in Figure 16.

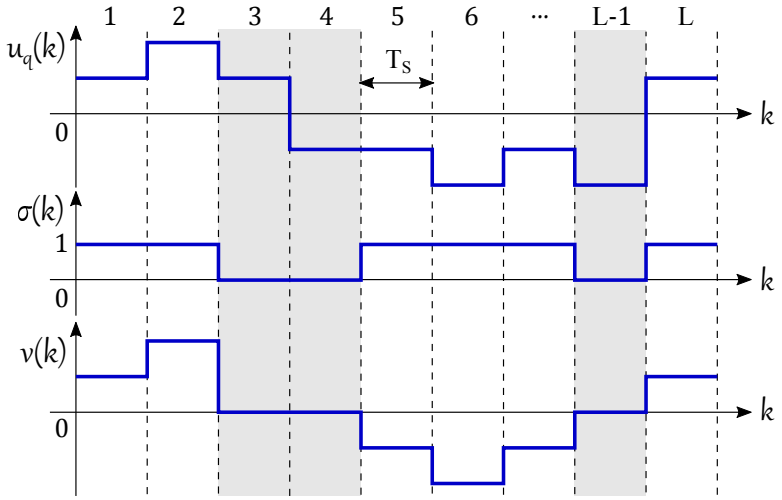



Figure 22: Operation of the trigger signal $\sigma(k)$ for the V0 type ETQ.

Finally, the design methods in Table 6 differ as well in the parameters that are given and the ones that they need to find. In particular, in methods M2 and M3 the parameters $\boldsymbol{\mu}$ and $\boldsymbol{s}^2$ are given. In these cases the means of the Gaussians

$\boldsymbol{\mu}$ are distributed uniformly along the range of the output signal $y(k)$. Thus, $\mu_i = y_{min} + (\Delta y/2) + \Delta y(i-1)$ for $i = 1, \ldots, n_G$, where $\Delta y = (y_{max} - y_{min})/n_G$ is the space between the means of two neighbor Gaussians; $y_{max}$ and $y_{min}$ are respectively the maximum and the minimum values of $y(k)$. In the case of the variances $\boldsymbol{s}^2$, it is assumed that $s_1^2 = s_2^2 = \ldots = s_{n_G}^2 = (\Delta y/3)^2$. This is actually the case illustrated in the upper graphic of Figure 17.

The design methods M2, M3 and M4 have constraint management strategies similar to the one used in M1. The virtual function $J_v(\boldsymbol{\theta})$ is the same, the one in Equation (87), but $p(\boldsymbol{\theta})$ is different in each case. In M2 the conditions that regulate the parameters $\boldsymbol{\mu}$ and $\boldsymbol{s}^2$ in $p(\boldsymbol{\theta})$ are absent, namely

$$p(\boldsymbol{\theta}) \;\; := \;\; \max[(\bar{\Lambda} - 1),\; -d^*,\; (\Psi - \Psi^*)]. \tag{92}$$

In the 2 Steps methods, M3 and M4, the function $p(\boldsymbol{\theta})$ is divided into two parts. For M3 the penalty $p(\boldsymbol{\theta})$ becomes

$$\text{Step 1:} \quad p_1(\boldsymbol{\theta}) \;\; = \;\; \max[(\bar{\Lambda} - 1),\; -d^*], \tag{93}$$

$$\text{Step 2:} \quad p_2(\boldsymbol{\theta}) \;\; = \;\; \Psi - \Psi^*, \tag{94}$$

each for one step of the design. For M4 the conditions that regulate $\boldsymbol{\mu}$ and $\boldsymbol{s}^2$ should be added as follows

$$\text{Step 1:} \quad p_1(\boldsymbol{\theta}) \;\; = \;\; \max[(\bar{\Lambda} - 1),\; -d^*], \tag{95}$$

$$\begin{aligned} \text{Step 2:} \quad p_2(\boldsymbol{\theta}) \;\; = \;\; & \max[(\Psi - \Psi^*),\; -s_{min}^2, \\ & (\mu_{max} - y_{max}),\; (y_{min} - \mu_{min})], \end{aligned} \tag{96}$$

In order to verify that these design methods work properly and to compare them with M1, additional numerical examples were performed. The plant, reference signal and settings of these examples are the same as the ones described in Section 4.3 including the form of the constant matrices $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, and $\boldsymbol{\mathcal{C}}$ in Equation (91). The simulations were performed again $N_{run} = 10$ times for each combination of $M$ and $\Psi^*$. After the simulations ended, the error system was fed with $u(k)$, Equation (90), to test the designed quantizers. They all work well, some examples of their output signals are shown in Figure 23. These examples show some interesting results.

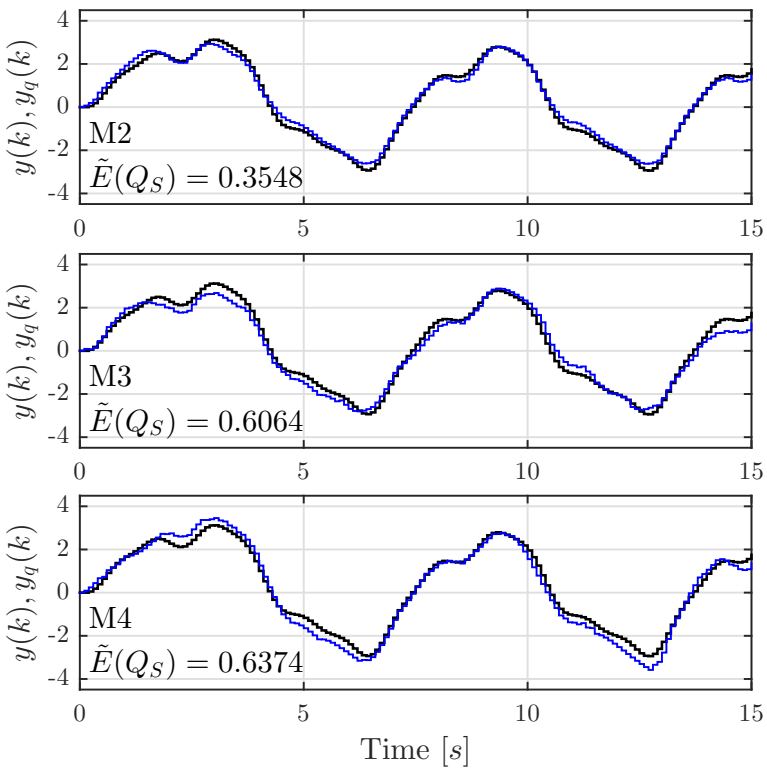



Figure 23: Output signals $y(k)$ (in black) and $y_q(k)$ (in blue) for the different design methods (M2, M3, and M4) when the system is fed with $u(k)$ and $Q_S$ is designed for $\Psi^* = 70\%$ and $M = 4$.

The considered design methods are compare to each other using statistical analysis techniques. First, the analysis of variance (ANOVA) is applied considering four groups of data. Each group corresponds to the performance indexes $\tilde{E}(Q_S)$ of a design method obtained from the simulation runs for a specific $M$ and $\Psi^*$. Thus, for each combination of $M = \{2, 4, 8, 16\}$ and $\Psi^* = \{70\%, 90\%\}$ a one-way ANOVA with a significance level $\alpha = 0.05$ is applied. The goal is to determine if there is some statistical difference among the performance indexes means of the design methods. For all the cases it was found that the null hypothesis is rejected. Then, in each case at least two design methods differ significantly. Following up after ANOVA, multiple comparison procedures such as Fisher's LSD and Tukye's test were applied. The conclusions taken from both procedures were the same. The first one is that the design methods in which $Q$ and $G$ are designed together (M1 and M2) outperform the ones in which they are designed separately (M3 and M4). This fact can be verified by comparing the output signal graphics and their performance indexes $\tilde{E}(Q_S)$ in Figure 23 and Figure 19.

Another observation is that the quantizers designed with the method M1 show the best performance among the other quantizers except for the case in which $M = 2$. For $M > 2$ the method M1 has the smallest mean in each combination of $M$ and $\Psi^*$. On the other hand, when $M = 2$ the method M2 gives the quantizers with the best performance. These results are exemplified in Table 7 where the minimum $\tilde{E}(Q_S)$ obtained with each design method for $\Psi^* = 90\%$ are shown. An explanation of this behavior is that in M2 (and M3) when the plant does not receive any $v(k)$ in a sampling time, it assumes $v(k) = 0$. Thus, if $M = 2$ there are only two values, different than 0, that $v(k)$ can take, then making $v(k) = 0$ on the side of the plant creates effectively a third quantization level. Accordingly, the error is reduced because $M$ is increased. This effect loses its appeal for $M > 2$, and maintaining the previous input when no $v(k)$ arrives to the plant showed to be more effective.

Regarding to the methods M3 and M4, that perform the design in two steps, the statistical analysis showed that for $M = 2$ the method M3 is better than M4 but when $M$ increases M4 showed to be superior. This result goes along with the theory because M4 has more variables and, in consequence, more flexibility.

Finally, it is worth to mention that due to the trade-off that happens between

Table 7: Minimum $\tilde{E}(Q_S)$ for $\Psi^* = 90\%$ for the different design methods and number of quantization levels $M$.

| Method | $M = 2$ | $M = 4$ | $M = 8$ | $M = 16$ |
|---|---|---|---|---|
| M1 | 0.5023 | **0.1216** | **0.0656** | **0.0454** |
| M2 | **0.3151** | 0.1469 | 0.0912 | 0.0845 |
| M3 | 0.7908 | 0.1443 | 0.1439 | 0.1461 |
| M4 | 1.4249 | 0.2815 | 0.2037 | 0.0703 |

the NUR and the errors' magnitude, it is important that the designer carefully selects the value of $\Psi^*$ having in account the system's needs and limitations.

## 4.5 ETQ for Multiple Input Signals

So far in this chapter the design of the ETQs have been carried out only for a single input signal $u(k)$. This is a severe limitations that affects the operation of the system. Ideally, an ETQ should should be designed independently of the input signal. However, a design method independent of the input signal may not exist. Thus, in an attempt to alleviate this limitation, in this section the design of the ETQ is carried out using a set of input signals. It is assumed that the considered input signals are bounded.

In this case the performance of the ETQ is evaluated using the error system depicted in Figure 18. The performance of the ETQ for multiple input signals $u(k)$ is evaluated using a generalization of the performance index in Equation (122) which used for the case of a single input signal. Thus, the performance index of ETQ for multiple $u(k)$ is defined as follows

$$\tilde{E}_m(Q_S) := \max_{i\in\{1,2,\ldots,n_u\}} \tilde{E}(u_i(k)) = \max_{\substack{i\in\{1,2,\ldots,n_u\}\\ k\in\{1,2,\ldots,L\}}} \operatorname{abs}\left(y_q(u_i(k)) - y_i(k)\right), \tag{97}$$

where, $n_u \in \mathbb{N}$ represents the considered amount of input signals. It is assumed that the considered input signals are independent from each other. The design problem is the same as in the case of a single input signal which is described in Section 4.2.5 with the difference that the performance index is the one in

Equation (97) and that the NUR generated by each signal should be less than $\Psi^*$. This design problem is solved using differential evolution.

In order to verify the effectiveness of the ETQs when they are designed for multiple $u(k)$ various numerical simulations were performed. Two design methods are considered, the 1 stepped original one M1 and the 2 stepped one M4. The settings of these examples are similar the ones in Section 4.3 for a single input signal. With the differences that a set of input signals is used instead of a single one, and that the performance index used in this case is the one in Equation 97. The considered set of input signals is the following

$$u_1(k) = 0.3\sin(6k) + 0.4\sin(k) + 0.3\sin(3k), \tag{98}$$

$$u_2(k) = 0.5\sin(k) + 0.5\sin(0.2512k), \tag{99}$$

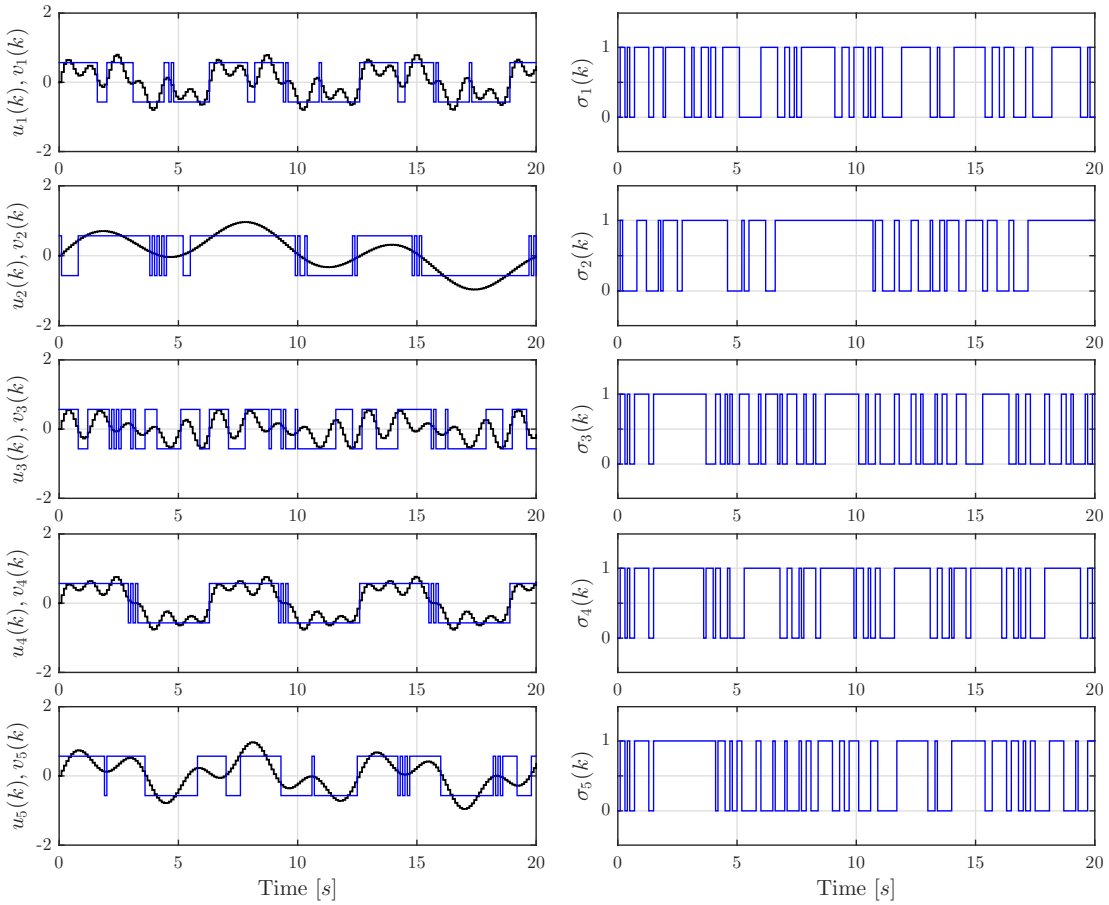
$$u_3(k) = 0.2\sin(4k) + 0.2\sin(k) + 0.3\sin(5k), \tag{100}$$

$$u_4(k) = 0.2\sin(3k) + 0.6\sin(k) + 0.2\sin(6k), \tag{101}$$

$$u_5(k) = 0.1\sin(0.2512k) + 0.4\sin(2.5119k) + 0.5\sin(k). \tag{102}$$

Thus, the considered plant is the one given in Equation 89, the sampling time is $t_S = 0.1$, and the evaluation interval is $L = 500$. The known parameters of $Q_S$ are the following $M = \{2,\ 8\}$, $n_G = 20$, $U = [-1, 1]$ and $\Psi^* = \{90\%,\ 70\%,\ 50\%\}$. The unknown parameters of $Q_S$ are the variables to be found by DE and they are arranged in a DE individual $\boldsymbol{\theta}$ as in Equation 91. The parameters of the DE algorithm are the same as before and the simulations are run $N_{run} = 10$ times. After the simulations ended and to verify that the designed quantizers work properly the error system was fed with the considered $u(k)$s. Some of the resulting signals are shown in Figure 24 and Figure 25.

Figure 24 shows the signal obtained from the ETQ designed using the design method M1 for $M = 2$ and $\Psi^* = 70\%$. The left column shows the considered input signals $u_i(k)$ $(i = 1, 2, \ldots, 5)$ in black and their respective quantized versions $v_i(k)$ $(i = 1, 2, \ldots, 5)$ in blue. In the right column there are shown the trigger signals $\sigma_i(k)$ $(i = 1, 2, \ldots, 5)$ for each $u(k)$. Figure 25a depicts the outputs, the ideal ones $y_i(k)$ $(i = 1, 2, \ldots, 5)$ in black and the ones due to quantization in blue $y_{q,i}(k)$ $(i = 1, 2, \ldots, 5)$. The left column show the outputs from the signals in Figure 24 for $\Psi^* = 70\%$. For comparison, in the right column there are shown the outputs for the quantizer designed for $M = 2$ and $\Psi^* = 90\%$. It is possible



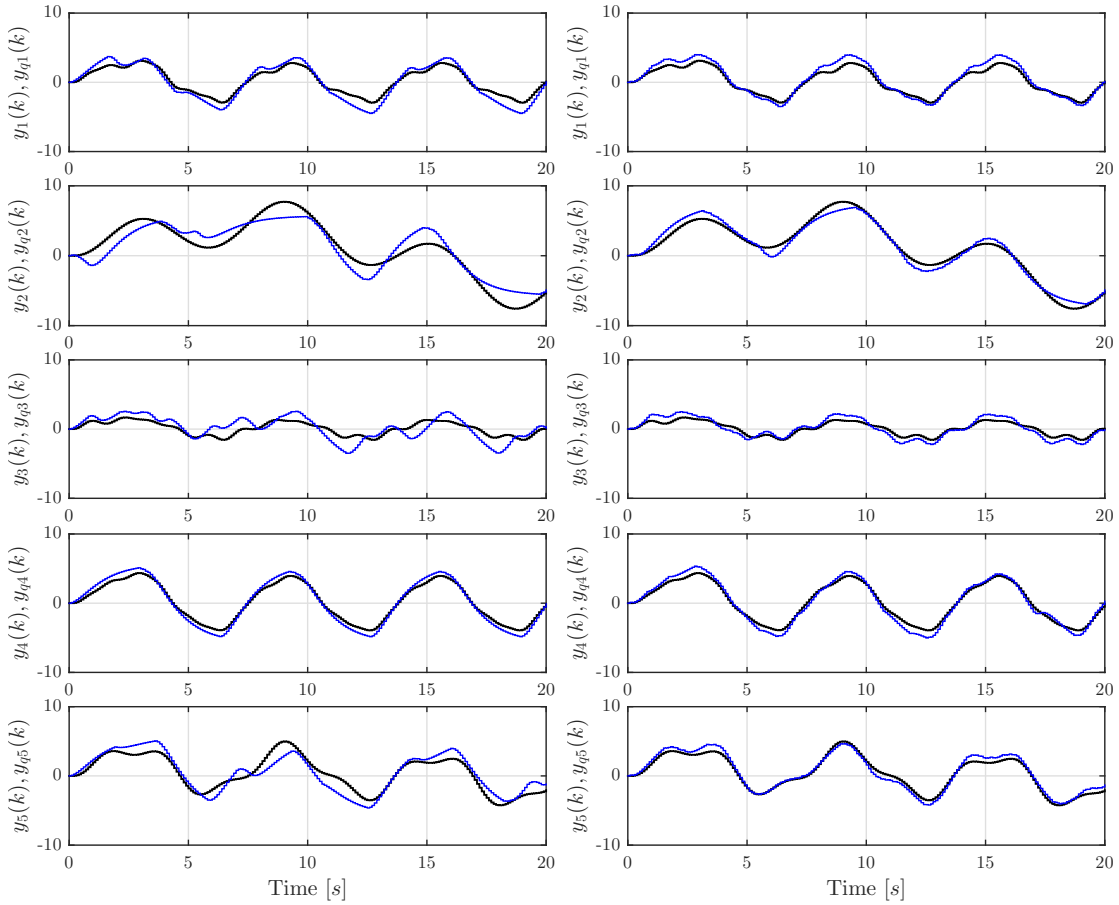

(a) Input signals.

(b) Trigger signals.

Figure 24: Signals resulting from exciting the ETQ designed with the M1 method, $M = 2$, $\Psi^* = 70\%$, and the set of considered $u(k)$. The signals in black are the input signals without quantization $u_i(k)$ $(i = 1, 2, \ldots, 5)$ and the ones in blue are the signals when event-triggered quantization is applied $v_i(k)$ adn $\sigma_i(k)$ $(i = 1, 2, \ldots, 5)$. Each row shows the inputs an outputs for one particular $u(k)$.

to see that the error between the output signals is relatively small and how the quantized outputs follow the ideal ones mostly well.

From these figures it is possible to infer that the ETQs design with M1 work relatively well. In the case of the ETQs designed for $M = 2$ using the 2 Steps design method M4, the errors are big making the quantizers not usable. In the




(a) Output signals for $\Psi^* = 70\%$.

(b) Output signals for $\Psi^* = 90\%$.

Figure 25: Output signals resulting from exciting the ETQs designed with the M1 method $M = 2$, $\Psi^* = \{70\%,\ 90\%\}$, and the set of considered $u(k)$. The signals in black are the ideal outputs $y_i(k)$ $(i = 1, 2, \ldots, 5)$ and the signals in blue are the ones obtained with event-triggered quantization $y_{q,i}(k)$ $(i = 1, 2, \ldots, 5)$. Each row shows the inputs an outputs for one particular $u(k)$.

case of $M = 8$ the errors are small for both design methods. To make a proper comparison among these design methods. A 1-way ANOVA test followed by the Tukey's test for a confidence level $\alpha = 0.01$ are run for each combination of $M$ and $\Psi^*$ being the factor the design methods. The results of these test show that for $M = 2$ the method M1 is significantly better than M4 in each case. For $M = 8$

there is only significant difference for $\Psi^* = 70\%$ being again M1 superior to M4. In this case M4 has better mean than M1 only for $\Psi^* = 90\%$, however there is no significant difference.

The main disadvantage of this design method is that the time used for the training can be high. It takes $n_u$ times longer to train this quantizer than in the case with a single $u(k)$.

## 4.6 Summary

In this chapter, the event-triggered dynamic quantizer is introduced in addition to four design methods. These design methods are based on the differential evolution algorithm, which is a reliable and easy to use metaheuristic. With several numerical examples the effectiveness of the proposed quantizer and its design methods are verified. The examples showed that method M1 produces the quantizers that have the best performance for $M > 2$ and method M2 for $M = 2$.

Some contributions of this chapter are the introduction of the event-triggered quantizer which is a new type of intelligent device that can be applied to real-word networked control systems and the use of event-trigger techniques which enable the intelligent control of dynamical systems. In particular, this devise employs the knowledge from previous input signals to reduce the traffic that the system puts in the network and codifies this knowledge in a function that is approximated using a weighted set of Gaussian functions.

A limitation of the presented design methods is that they perform the quantizer's design for a given input signal $u(k)$. Therefore, it is necessary to consider an event-triggered quantizer that is designed independently of the input signal or at least to consider the design for multiple input signals. A possible solution is to use a bigger amount of Gaussian functions or even another type of structures for the event-generator. Another limitation is that the differential evolution algorithm is not well suited for design problems with dimension higher than one hundred variables. Thus, if the dimension of the problem increases it would be necessary to look for other metaheuristics.

These limitations establish possible directions for future research. For instance, some future works are the theoretical analysis of the proposed quantizer to search for a design method independent of the input signal, and the quantizer's

experimental evaluation over real-world applications. Other research directions are the quantizer's implementation using different structures like artificial neural networks or Markov chain models, and the search for more suitable metaheuristics to perform the quantizer's design. Finally, another future work is the event-triggered quantizer's design taking into account the closed loop system and the control of the plant. A possibility is to combine the quantizer and the controller in a single device.

# 5. Design of Neural Network Quantizers (NNQs)

This chapter introduces a new type of noise-shaping quantizer that is constructed using neural networks and it is designed without the need of a model of the plant. Several variations of this quantizer are also proposed along side with a design method based on metaheuristics. With the aid of many simulations this quantizer and its design method proved their effectiveness. In addition, the variations of this quantizer are analysed and many interesting results are shown.

## 5.1 Introduction

So far in this study, the design of the quantizers was carried out using information from the plant, namely, they are model-based designs. Thus, the previous results cannot be directly applied to cases in which the model of the plant is unknown or cannot be used. Examples of these cases are that the plant cannot be identified, that the given model of the plant is not accurate, or it is just not practical to use the model due to its complexity. In addition, previous studies has found the expression of optimal dynamic quantizers for the cases of linear systems [4] and nonlinear systems [6]. These expressions are a function of the respective linear or nonlinear plant. A problem arises when the structure of the nonlinear plant is unknown, because in that case the structure of the optimal quantizer is unknown as well.

The problem of quantized systems with uncertainties in the plant has been considered by many researchers. A popular approach is to use robust control to make the systems resilient in a certain range against the plant's uncertainties [19, 70, 135]. Another well known approach is the use of adaptive control that provides on-line estimations of the systems' unknown parameters based on measurements [41, 42, 134, 80]. These studies focus mostly in the system stability and employ plant models along side with models of the uncertainties. Therefore, they are not very useful when the plant is unknown or the model is too complex. For these cases the data-driven control strategies are preferred. In the data-driven control the system's elements, such as controllers and observers, are implemented using artificial neural networks and the system is designed using series of input and outputs of the plant instead of a mathematical model [23, 113, 136, 60, 18]. The

used of neural networks in control systems is not uncommon. Several studies employ these networks to build controllers for linear or nonlinear systems with or without uncertainties [48, 49, 94, 96].

Accordingly, in this chapter the data-driven approach is considered for the design of feedback-type noise-shaping quantizers. These quantizers are constructed using feedforward neural networks and they are designed without the model of the plant, i.e., model-free design. It only needs a set of inputs of the plant, with their respective outputs, to perform the design. From here this quantizer will be referred to as *neural network quantizer* (NNQ). The selection of neural networks to perform this job is motivated by the fact that feedforward neural networks are very dynamic and that they can be used to represent any type of nonlinear function/system, in this sense they work as universal approximators [17, 50]. Some advantages of this approach are that it requires only a time series of input/output data from the plant and that it can be applied not only to linear but also to nonlinear plants. The neural networks can be trained to perform two types of tasks: regression and classification. For this reason, two types of NNQ are proposed, one based on regression and another based on classification. The difference between these quantizers will be explained in the following sections.

The design of these quantizers is carried out by optimizing the neural network's weights and biases. However, the problem is set up as a nonlinear and nonconvex optimization problem that cannot be solved by conventional neural network training techniques nor by conventional optimization methods. Therefore, this study uses the differential evolution (DE) metaheuristic algorithm to perform the quantizer's design.

This study was inspired by the work in [60] where the controller for a plant with unknown structure is designed using a set of plant's inputs and outputs. The main differences between these studies are that instead of implementing a controller the present study implements a quantizer, and that the neural network optimization is carried out using DE instead of techniques based on backpropagation.

This chapter is structured as follows, first the concept and basics of neural networks are presented. Second, the NNQ based on regression is introduced. Then, its design problem is formulated. Following, the NNQ based on classifica-

tion is described. With numerical examples the effectiveness of these quantizers and their design method are verified. After that, several design variations are considered in order to optimize the quantizer's performance. Finally, the case in which the quantizer is designed for multiple input signals is studied.

Part of this chapter's content was already published in [110], [111] and [105].

## 5.2 Feedforward Neural Networks

### 5.2.1 Structure

A neural network is a mathematical construction that is used to learn a mapping from a set of inputs to a set of target values. The network is composed by a series of interconnected nodes, the *neurons*, that perform mathematical transformations to their inputs. The neurons are arranged in *layers* and the information flows from one layer to another until the final layer provides the output of the network [9, 7, 33]. There are many types neural networks, in this chapter the fully connected feed-forward type will be considered to build the NNQs. An example of this type of network is shown in Figure 26.

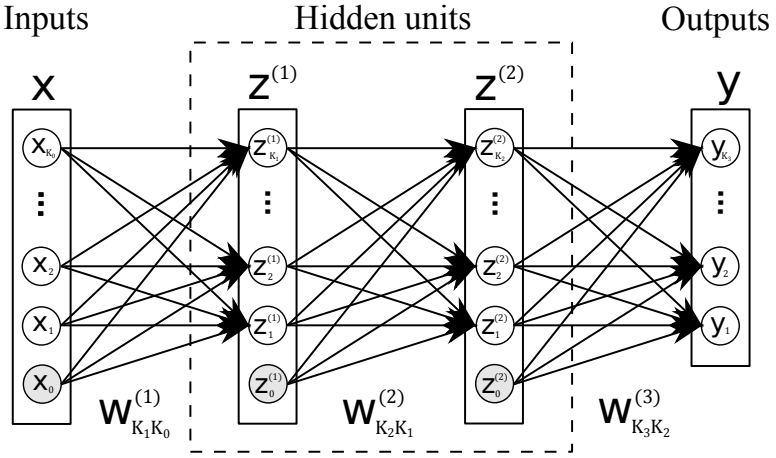



Figure 26: Fully connected 3 layered feed-forward neural network example.

The following elements can be recognized in the network, the input units $\mathbf{x} \in \mathbb{R}^{K_0}$, the output units $\mathbf{y} \in \mathbb{R}^{K_{n_L}}$, and the hidden units $\mathbf{z}^{(i)} \in \mathbb{R}^{K_i}$ $(i = 1, 2, \ldots, n_L - 1)$, where $n_L$ is the number of layers in the network. A useful

notation to describe a neural network is a vector such as $\boldsymbol{K} = [K_1, K_2, \ldots, K_{n_L}]$ in which its elements represent the number of neurons in each layer.

Each neuron performs a nonlinear transformation of a weighted summation of the previous layer outputs as follows

$$\mathsf{z}_j^{(l)} = \mathsf{h}\left(\sum_{i=0}^{K_l} \mathsf{w}_{ji}^{(l)} \mathsf{z}_i^{(l-1)}\right), \tag{103}$$

where $\mathsf{w}_{ji}^{(l)}$ represents the weight of the connection that goes from the $i^{\text{th}}$ neuron in layer $(l-1)$ to the $j^{\text{th}}$ neuron in layer $l$. Notice that here a simplified notation is used, where instead of having biases the units $\mathsf{x}_0 = 1$ and $\mathsf{z}_0^{(l-1)} = 1$ are included in the network. Then, because these elements are constants, their respective connection's weight $\mathsf{w}_{j0}^{(l)}$ serves as bias parameters. The weights of all the connections in the network are put together in a vector $\mathbf{w}$ called the *weights vector* that has dimension

$$n_{\mathsf{w}} = \sum_{i=0}^{n_L-1} (K_i + 1) K_{i+1}. \tag{104}$$

On the other hand, $\mathsf{h}(\cdot)$ represents the nonlinear transformation and is called *activation function*. There are many functions that serve as activation functions. The most commonly used for the hidden units are the logistic sigmoid, the hyperbolic tangent, and the rectified linear unit (ReLU). They are defined as follows

$$\text{sigm}(a) = \frac{1}{1 + \exp(-a)}, \tag{105}$$

$$\tanh(a) = \frac{1 - \exp(-2a)}{1 + \exp(-2a)}, \tag{106}$$

$$\text{ReLU}(a) = \max(a, 0). \tag{107}$$

These functions are depicted in Figure 27 for comparison.

As mentioned before, the neural networks can be designed to perform regression or classification. The activation function used in the output layer $\mathsf{h}_{out}(\cdot)$ depends on the type of task that the network performs. When it is designed for regression the activation function is just a linear function, being the identity function the most popular choice. When the network is designed for classification

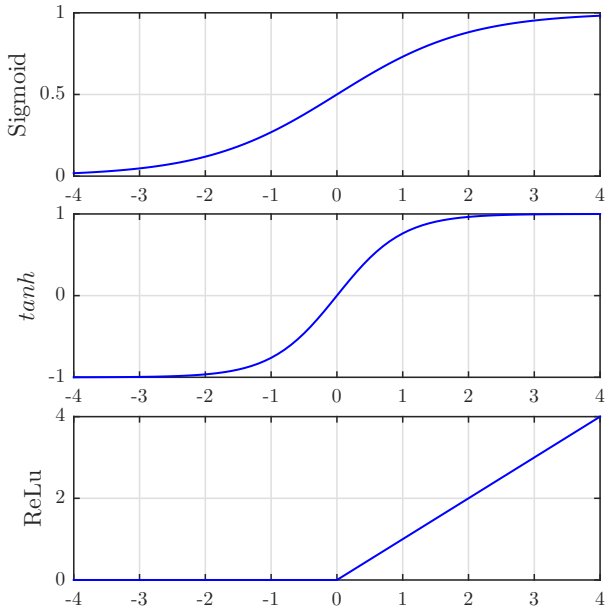



Figure 27: Hidden layers' activation function $\mathsf{h}(\cdot)$ comparison.

the softmax function is used. This function is the following

$$\text{softmax}(a_i) = \frac{\exp(a_i)}{\sum_{j=1}^{M} \exp(a_j)} \quad \text{for } i = 1, 2, \ldots, K_{n_L}. \tag{108}$$

Finally, the output of the network $\mathsf{y}(k)$ is calculated by propagating the inputs from layer to layer until the output layer is reached. For example, in the case of $n_L = 3$ the overall *network function* is given by

$$\mathsf{y}_l(\mathbf{x}) = \mathsf{h}_{out}\left(\sum_{k=0}^{K_2} \mathsf{w}_{kl}^{(3)} \mathsf{h}\left(\sum_{j=0}^{K_1} \mathsf{w}_{kj}^{(2)} \mathsf{h}\left(\sum_{i=0}^{K_0} \mathsf{w}_{ji}^{(1)} \mathsf{x}_i\right)\right)\right), \tag{109}$$

for $l = 1, 2, \ldots, n_L$.

A neural network is considered to be deep when it has four or more layers. Deep neural networks are a relatively new development in machine learning field that has greatly improved the performance of images and speech recognition [62, 61].

### 5.2.2 Initialization

The learning resulting from the training of a deep neural network depends highly on the initial weights of the network because many of the learning techniques are in essence local searches. Therefore, it is very important to initialized the network's weights appropriately [130, 2].

There are several ways to initialize the neural networks to perform the training. The most common method is the uniformly random initialization where random values sampled from a certain interval using a uniform probability distribution are assigned to the weights and biases of the network. The initialization intervals are selected according to the networks characteristics but they are usually small and close to zero. Popular ones are the intervals $[-1, 1]$ or $[-0.5, 0.5]$.

Another very popular type of initialization was developed in [32] by Glorot and Bengio. This method is know are Xavier Uniform initialization (from Xavier Glorot). In this method the weights of each layer in the network are initialized using random uniform sampling in a specific interval

$$\mathbf{w}_i \sim U[-l_i, \quad l_i], \quad \mathbf{b}_i = \mathbf{0} \quad \text{for} \quad i = 1, 2, \ldots, n_L, \tag{110}$$

where, $\mathbf{w}_i$ and $\mathbf{b}_i$ represent the weights and biases of the $i$th layer, receptively. Notice that the biases are initialized to zero. The limits of the interval are given by $l_i$ which is a function of the number of neurons of the considered layer $K_i$, the number of neurons in the previous layer $K_{i-i}$ and the hidden layers activation function $\mathsf{h}$. These limits are the following

$$l_i = \frac{4\sqrt{6}}{\sqrt{K_{i-1} + K_i}} \quad \text{for} \quad \mathsf{h}(a) = \text{sigm}(a), \tag{111}$$

$$l_i = \frac{\sqrt{6}}{\sqrt{K_{i-1} + K_i}} \quad \text{for} \quad \mathsf{h}(a) = \tanh(a), \tag{112}$$

In [43], He et al. extended this method for the ReLU activation function as follows

$$l_i = \frac{\sqrt{6}}{\sqrt{K_i}} \quad \text{for} \quad \mathsf{h}(a) = \text{ReLU}(a), \tag{113}$$

## 5.3 Regression Based NNQ

### 5.3.1 System Description

This study considers the system depicted in Figure 28.

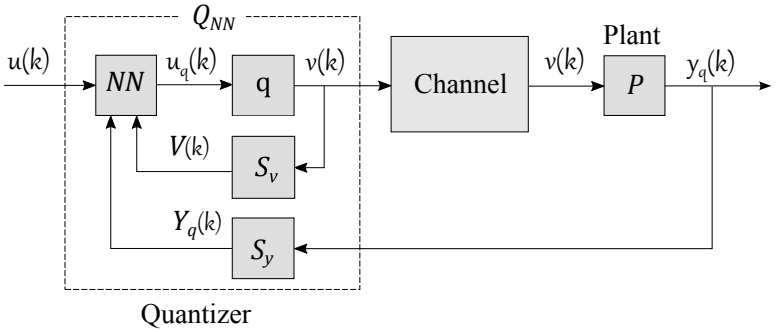



Figure 28: Considered system with a neural network quantizer $Q_{NN}$.

This system is composed of a quantizer $Q_{NN}$, a communication channel that has no looses or delays, and a plant $P$ with unknown structure.

The plant is represented by the following single-input-single-output (SISO) general model

$$P : \begin{cases} \boldsymbol{x}(k+1) & = f\left(\boldsymbol{x}(k), v(k)\right), \\ y(k) & = g\left(\boldsymbol{x}(k)\right), \end{cases} \tag{114}$$

where $k \in \{0\} \cup \mathbb{N}$ is the discrete time, $\boldsymbol{x} \in \mathbb{R}^{n_P}$ is the state vector with initial value $\boldsymbol{x}(0) = \boldsymbol{x}_0$, $v \in \mathbb{R}$ is the input signal, and $y \in \mathbb{R}$ is the output signal. The functions $f : \mathbb{R}^{n_P} \times \mathbb{R} \to \mathbb{R}^{n_P}$ and $g : \mathbb{R}^{n_P} \to \mathbb{R}$ are in general nonlinear mappings. It is assumed that $f$ and $g$ are continuous and smooth.

Another assumption is that, although the model is unknown, the plant is stable and it is possible to feed it with inputs and measure its outputs. Then, a time series of inputs and outputs of the plant will be available. These time series are represented as follows

$$\boldsymbol{U} = [u(1), u(2), \ldots, u(n_s)]^\top, \tag{115}$$

$$\boldsymbol{Y} = [y(1), y(2), \ldots, y(n_s)]^\top, \tag{116}$$

where $n_s$ is length of the time series, namely, the amount of samples. Notice that $y(k)$ $(k = 1, 2, \ldots, n_s)$ represents the output of the plant $P$ when $u(k)$ is applied directly to it, i.e., $v(k) = u(k)$.

The quantizer $Q_{NN}$, shown in Figure 28, is composed of a neural network $NN$, a static quantizer q, and a couple of memories, $S_v$ and $S_y$. It is represented as follows

$$Q_{NN} : \begin{cases} u_q(k) &= \Gamma\left[u(k), \boldsymbol{V}(k), \boldsymbol{Y}_q(k)\right], \\ v(k) &= \mathrm{q}\left[u_q(k)\right], \end{cases} \tag{117}$$

where $u \in \mathbb{R}$ is the input signal, $u_q \in \mathbb{R}$ is the output of $NN$, and $v \in \{\pm\frac{d}{2}, \pm 2\frac{d}{2}, \cdots, \pm\frac{M}{2}\frac{d}{2}\}$ is the output of $Q_{NN}$, the quantized input signal.

The signals $\boldsymbol{V}(k)$ and $\boldsymbol{Y}_q(k)$ are, respectively, the outputs of the memories $S_v$ and $S_y$. They are time series of past values of the quantized inputs $v(k)$ and the outputs of the plant $y_q(k)$, as follows

$$\boldsymbol{V}(k) = [\ v(k-1),\ v(k-2), \ldots,\ v(k-n_V)]^\top, \tag{118}$$

$$\boldsymbol{Y}_q(k) = [y_q(k-1), y_q(k-2), \ldots, y_q(k-n_Y)]^\top, \tag{119}$$

where $n_V$ and $n_Y$ are the dimensions of the memories.

The static quantizer q[·] receives the continuous output of the neural network $u_q(k)$ and rounds it to the nearest discrete value to generate $v(k)$. It has two parameters: one is the number of quantization levels $M \in \mathbb{N}$ and the other is the quantization interval $d \in \mathbb{R}$ with $d > 0$. Figure 29 shows an example of the static quantizer with $M = 4$.

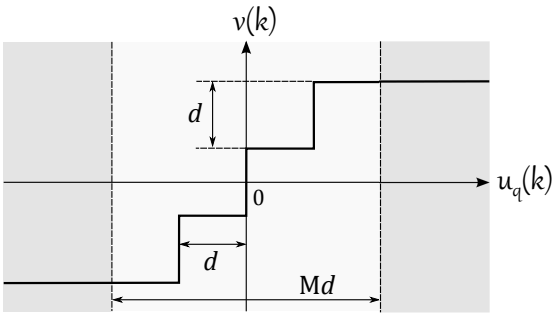



Figure 29: Example of a static quantizer q[·] ($M = 4$).

The nonlinear function $\Gamma : \mathbb{R} \times \mathbb{R}^{n_V} \times \mathbb{R}^{n_Y} \to \mathbb{R}$ represents the neural network function. Inside $Q_{NN}$ the neural network implements the dynamics of the quantizer. The inputs of $NN$ are $u(k)$, $\boldsymbol{V}(k)$ and $\boldsymbol{Y_q}(k)$, and its output is the signal $u_q(k)$, they are represented as follows

$$\mathbf{x}(k) = \left[u(k),\ \boldsymbol{V}^\top(k),\ \boldsymbol{Y_q}^\top(k)\right]^\top, \tag{120}$$

$$u_q(k) = \mathbf{y}(k). \tag{121}$$

Accordingly, the dimension of the neural network input $\mathbf{x}(k)$ is $K_0 = 1 + n_V + n_Y$ and the first equation of (117) can be written as $\mathbf{y}(k) = \Gamma[\mathbf{x}(k)]$. Initially, the logistic sigmoid, Equation (105), is used as activation function for the hidden units, and because the regression type of network is considered is this section, the output units activation function is the identity function $\mathsf{h}_{out}(a) = a$. Notice that in this case $K_{n_L} = 1$ since there is only one output $u_q(k)$.

In this study it is assumed that $M$, $n_V$, $n_Y$ and $\boldsymbol{K}$ are given. Thus the design parameters of $Q_{NN}$ are $\mathbf{w}$ and $d$.

### 5.3.2 Training Data

The sampled data available to perform the training of the neural network are $\boldsymbol{U}$ and $\boldsymbol{Y}$. Traditionally, in order to train a neural network it is necessary to have a series of sampled inputs and target values that match the inputs and outputs of the network. However, in the considered case the available target values $\boldsymbol{Y}$ do not correspond to the outputs of the network. Instead, the neural network's output $u_q(k)$ is transformed by q and the unknown plant to the output $y_q(k)$ that can be compared with the target values as Figure 30 shows.

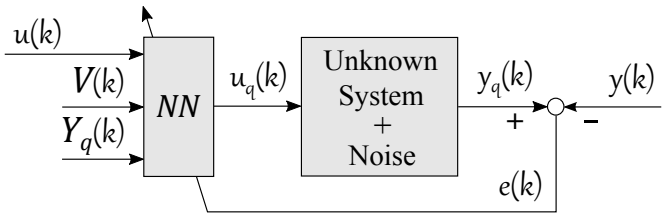



Figure 30: Supervised learning of the quantizer.

For this reason the conventional methods based on error backpropagation cannot be used to train the network. A solution to this problem is to perform

supervised training of the network using some other method like metaheuristics to minimize an error function.

### 5.3.3 Performance Index

Since the neural network cannot be trained by conventional methods, the design of $Q_{NN}$ is carried out considering the error system depicted in Figure 31.

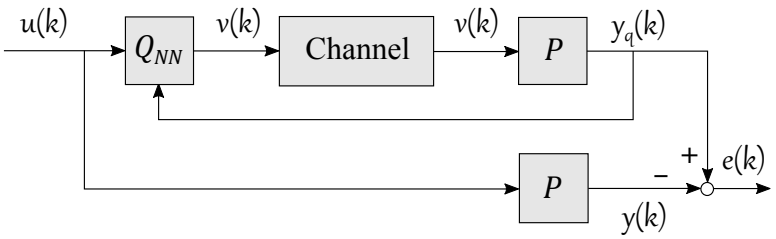



Figure 31: Error system considered.

This system is composed of two branches. In the lower branch the input $u(k)$ is applied directly to the plant $P$ that produces the ideal output $y(k)$. In the upper branch the effects of quantization are considered and $u(k)$ is applied to the quantizer $Q_{NN}$ that generates the quantized signal $v(k)$ that is applied to the plant. The output of the plant in this case is represented by $y_q(k)$, and the difference $e(k) = y_q(k) - y(k)$ is the error signal.

The performance of $Q_{NN}$ is evaluated using the sum-of-squares error function, that here works as the *performance index* that was defined in previous chapters. This function is defined as follows

$$E(\mathbf{w}, d) = \frac{1}{2} \sum_{k=0}^{n_s} [y_q(u(k), \mathbf{w}, d) - y(k)]^2 \tag{122}$$

where $u(k)$ is used to build $\mathbf{x}(k)$ along side with $\boldsymbol{V}(k)$ and $\boldsymbol{Y}_q(k)$ that are generated dynamically. This error function is used in the machine learning field when the neural networks are trained for regression, and the samples are independent and identically distributed [9].

It is desired to make the performance index as small as possible in order to maintain the output error low. Then, the design of $Q_{NN}$ is set up as an optimization problem in which the performance index is minimized.

### 5.3.4 Design Problem

The design problem can be formulated as:

Suppose that $M$, $\boldsymbol{U}$, $\boldsymbol{Y}$, $\boldsymbol{K}$, $n_V$, and $n_Y$ are given. Then, find the parameters of $Q_{NN}$: $\mathbf{w}$, and $d$ which minimize $E(\mathbf{w}, d)$, under the condition that:

i The quantization interval is positive ($d > 0$).

This design problem is nonlinear and nonconvex. Thus it cannot be solved using gradient based optimization methods like linear programming or quadratic programming. Moreover, conventional neural network training techniques based on error backpropagation cannot be used neither due to the structure of the system, as it was shown previously. Therefore, alternative optimization methods should be used.

In this regard the metaheuristics stand out from the available options because of their flexibility and big variety of implementations [89]. In particular, the differential evolution (DE) metaheuristic algorithm is used to perform the design of $Q_{NN}$. This choice is justified by the fact that DE has proved to be effective in the training of feedforward neural networks [52] and that it has shown a very good performance in the design of dynamic quantizers [106, 107, 108]. The version of DE implemented here is the DE/best/1/bin strategy, which is described in Algorithm 2.

Since the design parameters of $Q_{NN}$ are $\mathbf{w}$ and $d$, an individual for the DE algorithm will have the following form $\boldsymbol{\theta} = [d \;\; \mathbf{w}]^\top$ with dimension $n = 1 + n_{\mathbf{w}}$. From these parameters, the weights vector $\mathbf{w}$ is not affected by any constraint, but the quantization interval $d$ should always be positive $d > 0$.

As mentioned in previous chapters, DE has no direct way to handle the constraints of the optimization problem. Thus, the constraint management method described in Section 2.2 is used. This method transforms the constrained optimization problem into the following unconstrained one.

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}}\, J(\boldsymbol{\theta}), \qquad \text{for} \qquad J(\boldsymbol{\theta}) := \begin{cases} J_v(\boldsymbol{\theta}) & \text{if } p(\boldsymbol{\theta}) < 0, \\ p(\boldsymbol{\theta}) & \text{otherwise,} \end{cases} \tag{123}$$

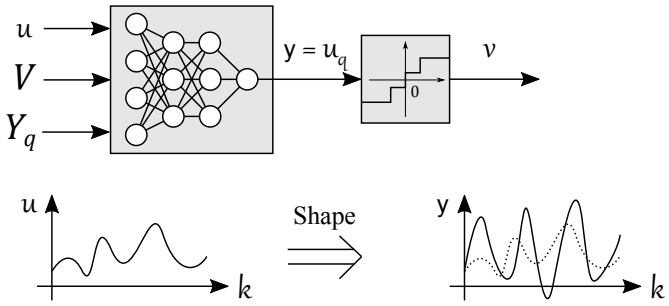
$$J_v(\boldsymbol{\theta}) := \arctan[E(\boldsymbol{\theta})] - \pi/2, \tag{124}$$

$$p(\boldsymbol{\theta}) := -d, \tag{125}$$

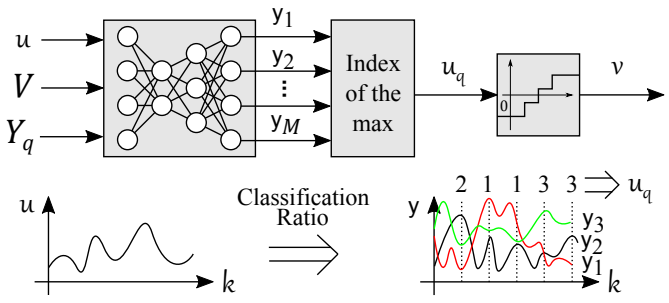
where $E(\boldsymbol{\theta})$ correspond to (122). This condition assures that $d$ is positive.

## 5.4 Classification Based NNQ

The neural network quantizer based on classification $Q_{NNC}$ shares the same principles as the one based on regression $Q_{NNR}$. The main difference between these quantizers is in the neural network's structure. In $Q_{NNR}$ the network has only one output that shapes the input signal, i.e., the network is trained to perform regression. In $Q_{NNC}$ the network has as many outputs as the considered amount of quantization levels $M$. Each output represents the probability that a given input is matched with a specific quantization level, i.e., the network is trained for classification. Figure 32 helps to clarify this difference.




(a) Regresion based approach.




(b) Classification based approach.

Figure 32: Difference between the neural network quantizer based on regression and the one based on classification.

The system in which $Q_{NNC}$ is implemented is the same as $Q_{NNR}$, the one in Figure 28. This is quantizer is also represented by Equation (117), with the

memories are described in Equations (118) and (119). The static quantizer q(·), however, is not a conventional one. Its input $u_q(k)$ represents a set of indexes, each of which makes reference to a specific quantization level. Thus, q is adapted to match each index to the corresponding quantization level as Figure 33 shows. This quantizer is defined by the number of quantization levels $M \in \mathbb{N}$ and the quantization interval $d \in \mathbb{R}_+$.

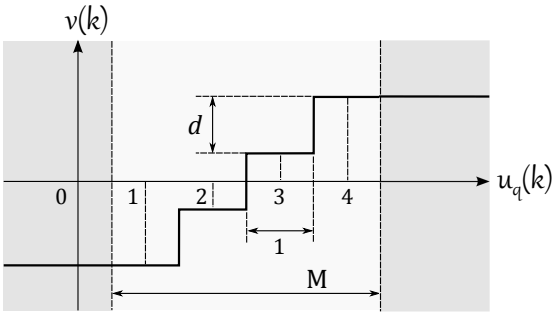



Figure 33: Example of static quantizer q[·] adapted for the NNQ based on classification ($M = 4$).

The dynamics of $Q_{NNC}$ are implemented with a fully connected feedforward neural network that is trained to perform classification [9, 33]. The inputs of this network are the same as in the previous case $\mathbf{x}(k) = \left[u(k),\ \mathbf{V}^\top(k),\ \mathbf{Y_q}^\top(k)\right]$ and the hidden units activation function $\mathsf{h}(\cdot)$ is the logistic sigmoid in Equation (105). However, the outputs $\mathbf{y}(k)$ are different. Since the network is trained for classification, the output layer's activation function $\mathsf{h}_{out}(\cdot)$ is the softmax function in Equation (108). Thus, each output of the network $\mathsf{y}_i(k)$ is associated with one quantization level. Indeed, because of this $\mathsf{h}_{out}(\cdot)$, each output represents the probability that a given input is classified into a specific quantization level. Therefore, the quantization level with the biggest probability is selected to be the network's output, and $\Gamma(\cdot)$ is modified as follows

$$\Gamma : u_q(k) = \underset{i \in \{1,2,\ldots,M\}}{\arg\max}\ \mathsf{y}_i(k). \tag{126}$$

As in the case of $Q_{NNR}$, the parameters $M$, $n_V$, $n_Y$ and $\boldsymbol{K}$ are considered given by the designer. Then, the design parameters of $Q_{NNC}$ are $\mathbf{w}$ and $d$.

Finally the same principles and strategies described in Sections 5.3.2, 5.3.3 and 5.3.4 are apply to $Q_{NNC}$. Thus, the design problem of $Q_{NNC}$ is formulated

as well as an optimization problem that is solved using DE. It is important to notice that the cost function used to train $Q_{NNC}$ is the same as the one use for $Q_{NNR}$, the *sum-of-squares* error function in Equation (122). Usually, a neural network that is trained for classification would be using the *cross-entropy* error function as cost function [122], but due to the situation described in Section 5.3.3 the sum-of-squares error functions is employed instead.

## 5.5 Numerical Examples

To verify that the neural network quantizer $Q_{NN}$ in its two variations and its design method work properly, several numerical simulations were performed. In these simulations the following discrete, nonlinear and stable plant is used

$$P : \begin{cases} \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}(k)) + f_3(\boldsymbol{x}(k))u(k) \\ f_2(\boldsymbol{x}(k)) + f_3(\boldsymbol{x}(k))u(k) \end{bmatrix}, \\ \\ y(t) = 1.45x_1(k) + x_2(k), \end{cases} \tag{127}$$

$$\begin{aligned} f_1(\boldsymbol{x}) &= 0.8x_1 - 0.4x_2 + 0.4e^{-|x_2|}\cos^3(x_1), \\ f_2(\boldsymbol{x}) &= 0.6x_2 + 0.4e^{-|x_1|}\left|\cos(x_1)\right|^{\frac{1}{2}}, \\ f_3(\boldsymbol{x}) &= 0.01 + 0.01((x_1)^4 + 0.1)^{-1}. \end{aligned}$$

The sampling time is $t_S = 0.1[s]$ and the initial state is $\boldsymbol{x}_0 = [0.1,\ -0.2]^\top$. The input signal used in the examples is the following

$$u(k) = 0.3\sin(6k) + 0.4\sin(k) + 0.3\sin(3k), \tag{128}$$

and the evaluation interval is $L = 1000$, which implies that amount of samples taken is $n_s = 1000$.

The quantizers are constructed with $n_Y = n_V = 5$, $M = \{2,\ 8\}$ and neural networks with $n_L = \{2,\ 4\}$. Given the size of the memories and the dimension of $u(k)$ all the networks have inputs with dimension $K_0 = 11$. The neural networks' structure depends on the type of quantizer and $M$. Table 8 summaries the structure of the quantizers used in the simulations. For the regression case (R) the network's structure and the dimension of $\mathbf{w}$ ($n_{\mathbf{w}}$) are independent of $M$. This

Table 8: Considered neural networks' structure.

| Type | $M$ | $\boldsymbol{K}$ | $n_L$ | $n_{\mathsf{w}}$ | $n$ |
|---|---|---|---|---|---|
| R | $\{2,\ 8\}$ | $[10,\ 1]$ | 2 | 132 | 133 |
| | | $[10,\ 10,\ 10,\ 1]$ | 4 | 352 | 353 |
| C | 2 | $[10,\ 2]$ | 2 | 142 | 143 |
| | | $[10,\ 10,\ 10,\ 2]$ | 4 | 362 | 363 |
| | 8 | $[10,\ 8]$ | 2 | 208 | 209 |
| | | $[10,\ 10,\ 10,\ 8]$ | 4 | 428 | 429 |

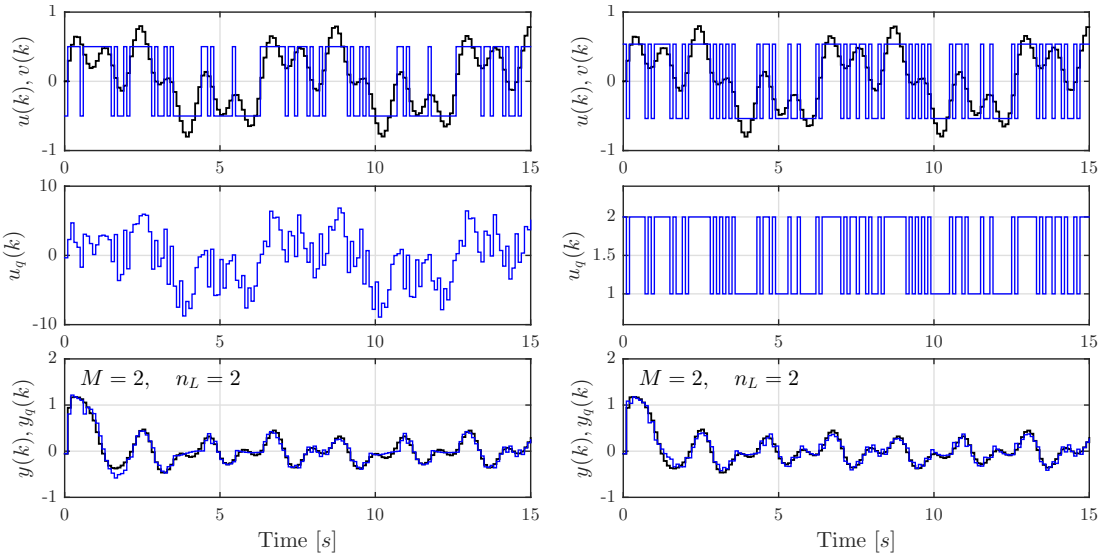
is not the case for the classification type of quantizer (C). Table 8 also shows a comparison among the $n_{\mathsf{w}}$ of each network.

The control parameters of DE are $N = 500$, $k_{max} = 2000$, $F = 0.6$, and $H = 0.9$. The simulations were performed $N_{run} = 50$ times for each considered case. Then, since the individuals have the form $\boldsymbol{\theta} = [d \ \ \mathbf{w}]^\top$ the dimensions of the optimization problems $n$ will be the ones shown is the last column of Table 8. Looking at Table 8 it is possible to see that $Q_{NNC}$ has more parameters than $Q_{NNR}$, this is a factor that influences the performance of the proposed design method.

The DE individuals are initialized as follows. The first element $d$ is uniform randomly sampled from the interval $[0, 1]$. The following elements, the network weights and biases, are initialized using the uniform random and the Xavier uniform initialization methods, described in Section 5.2.2. For the uniform random method the initialization interval is $S_0 = [-1, 1]^n$. In the implementation of the Xavier method a small modification is made. Instead of initializing the biases to zero there are sampled uniformly in an interval close to zero. For these numerical examples the interval $[-0.5, 0.5]$ is used. This is done because if all individuals have the same elements equal to zero, the mutation and crossover processes will not generate values different than zero for these elements.

After running the DE algorithm $N_{run}$ times for each considered case, the quantizers $Q_{NN}$ with the lowest $E(\mathbf{w}, d)$ are selected to be the optimal quantizers. Then, in order to test these quantizers, the error system in Figure 31 is fed with

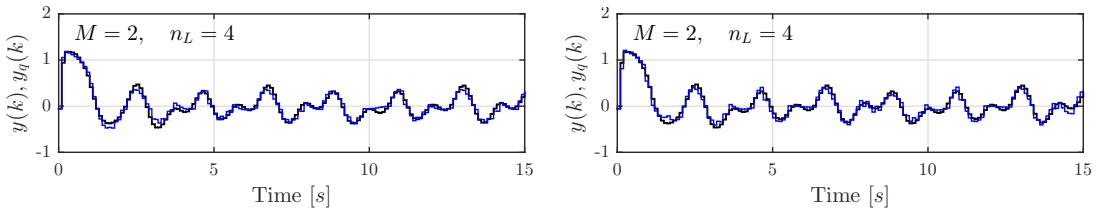
the input signal $u(k)$ for each case. It results that all the quantizers work properly and show good performance. For instance, Figure 34 depicts the signals resulting from applying $u(k)$ to the system with the quantizers designed for $M = 2$ and $n_L = 2$. This figure shows that the output signals obtained by quantization $y_q(k)$ follow the ideal output signal $y(k)$ pretty well and that the error between them is small in both cases. Also, the inputs of the static quantizers $u_q(k)$ are shown for comparison.




(a) $Q_{NNR}$. (b) $Q_{NNC}$.

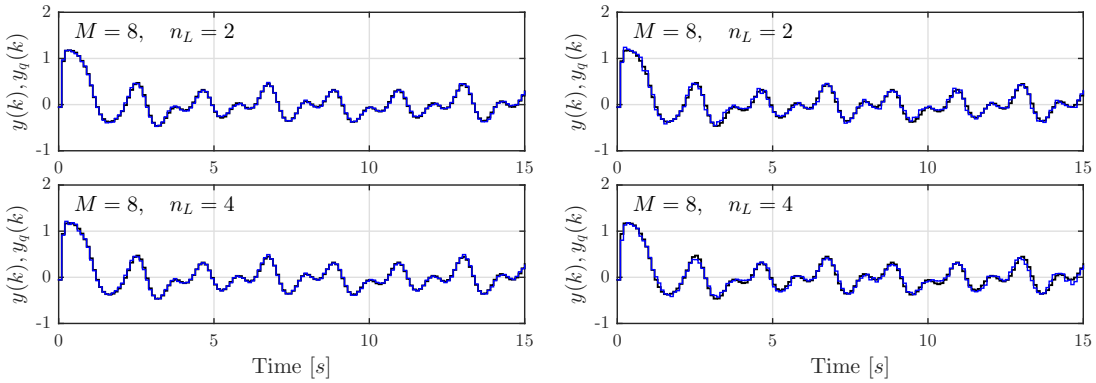
Figure 34: Signals resulting from applying $u(k)$ to the system with the NNQs designed for $M = 2$ and $n_L = 2$. The black lines represent the signals without quantization ($u(k)$, $y(k)$) and the blues ones are the signals when quantization is applied ($v(k)$, $u_q(k)$, $y_q(k)$).

To further validate this observation, in Figure 35 there are shown the output signals of the system where the NNQs were designed for $M = 2$ and $n_L = 4$, and in Figure 36 the ones for $M = 8$, $n_L = 2$ and $n_L = 4$.




(a) $Q_{NNR}$. (b) $Q_{NNC}$.

Figure 35: Output signals $y_q(k)$ (blue) and $y(k)$ (black) resulting from applying $u(k)$ to the system with $Q_{NN}$ designed for $M = 2$ and $n_L = 4$.




(a) $Q_{NNR}$. (b) $Q_{NNC}$.

Figure 36: Output signals $y_q(k)$ (blue) and $y(k)$ (black) resulting from applying $u(k)$ to the system with $Q_{NN}$ designed for $M = 8$, $n_L = 2$ (upper graphic) and $n_L = 4$ (lower graphic).

### 5.5.1 Regression and Classification Based NNQ Comparison

The minimum values of the performance indexes, Equation (122), found by DE, are listed in Table 9. In addition, this table lists the average performance indexes and their standard deviation. The values in this table are divided according to their $M$, initialization method, $n_L$ and type (regression or classification). There are two initialization methods implemented: uniform random (Urand) and Xavier.

It is difficult to draw conclusions from this table by simple observation. For

Table 9: $E(Q_{NN})$ analysis for $\mathsf{h} = \text{sigmoid}$ ($N_{run} = 50$).

| $M$ | Init. | $n_L$ | Type | Min. | Avg. | Std. Dev. |
|---|---|---|---|---|---|---|
| 2 | Urand | 2 | R | 3.73724 | 4.32987 | 0.48300 |
| | | | C | 3.66946 | 4.27038 | 0.34762 |
| | | 4 | R | 3.66764 | 4.22516 | 0.45819 |
| | | | C | 3.54773 | 4.30158 | 0.49296 |
| | Xavier | 2 | R | 3.42879 | 4.15830 | 0.36054 |
| | | | C | 3.53307 | 4.09696 | 0.35038 |
| | | 4 | R | 3.65081 | 4.10635 | 0.35909 |
| | | | C | 3.63822 | 4.04066 | 0.29597 |
| 8 | Urand | 2 | R | 0.17825 | 0.20622 | 0.01612 |
| | | | C | 0.91201 | 2.91111 | 1.54333 |
| | | 4 | R | 0.22911 | 0.32243 | 0.14041 |
| | | | C | 0.81667 | 2.81630 | 1.43382 |
| | Xavier | 2 | R | 0.20424 | 1.90045 | 1.22825 |
| | | | C | 0.93284 | 2.85580 | 1.44467 |
| | | 4 | R | 0.24762 | 0.85188 | 1.03043 |
| | | | C | 1.00016 | 2.61201 | 1.06714 |

example, looking at the minimum values of $E(Q_{NN})$ in the case of $M = 2$, it is possible to says that $Q_{NNC}$ have better performance (smaller $E(Q_{NN})$) than $Q_{NNR}$ in most cases. The average values not always corroborate this observation. For $M = 8$, $Q_{NNR}$ has the smallest $E(Q_{NN})$ in each case. However, there is no evidence that there is significant difference between these types of quantizer. Therefore, the analysis of variance (ANOVA) is used to check if there are significant differences among these values.

Because there are many factors that influence $E(Q_{NN})$ the a one-way ANOVA analysis may not be appropriate. Instead, the 3-way ANOVA (ANOVA with 3 factors) is used. The considered factors are Type, initialization method (Init.) and number of layers $n_L$. The categories of each factor are known as elements. For instance, the elements of the factor Type are R (regression) and C (classification).

The $M$ is not taken as a factor, because $M = 8$ clearly gives smaller $E(Q_{NN})$s than $M = 2$. Then, it is not necessary to check which one gives better results. The considered significance level is $\alpha = 0.05$. The goal is to determine if there is some statistical difference among the $E(Q_{NN})$'s means of the design methods.

The ANOVA test tells if there are significant differences among sets of data. When doing 3-way ANOVA it is possible to see not only if there is significant difference among elements of a factor but also among combinations of elements of different factors. In this particular case, it will tell if there is significant difference between $Q_{NNR}$ and $Q_{NNC}$, and also it will tell if there are differences among the combinations of the quantizer types and the initialization methods. Then, the 3-way ANOVA test is run separately for $M = 2$ and $M = 8$. For the case of $M = 2$ the significant difference is found only for the initialization method. For the case of $M = 8$ the significant difference is found for all the factors and the combinations of them with exception of the combination of the quantizer type and $n_L$. The details of these analyzes are shown in Table 17 in Appendix C.

The ANOVA test only tells if the null hypothesis that there is no significant difference among the means is rejected or not. It does not tell which element of a factor is better. For this purpose, comparison procedures such as Fisher's LSD and Tukey's test should be used. In this study the Tukey's test is used to perform the comparison. A summary of this test is shown in Table 10 for the individual factors. The results for the combination of factors are shown in Table 18 for $M = 2$ and in Table 19 for $M = 8$ in Appendix C.

In Table 10 the means that do not share a letter are significantly different. In addition, the means of the elements of each factor are decreasing from top to bottom. Thus, the following conclusions are extracted from the table. First, for $M = 2$ there is no significant difference between $Q_{NNR}$ and $Q_{NNC}$ nor between the number of layers $n_L = 2$ and $n_L = 4$. However, there is significant difference between the initialization methods, and the Xavier method outperforms the Urand method. Second, for $M = 8$ there is significant difference between the elements of each single factor. Then, $Q_{NNR}$ is better than $Q_{NNC}$, the Urand initialization method shows better performance than the Xavier method and the quantizers with $n_L = 4$ have better performance than the ones with $n_L = 2$.

These results are interesting because of the differences found if both cases of

Table 10: Tukey pairwise comparison 3-way ANOVA for $\mathsf{h}$ = sigmoid and single factors. Grouping Information Using the Tukey Method and 95% confidence. *Means that do not share a letter are significantly different.*

| $M$ | Factor | | N | Mean | Grouping | |
|---|---|---|---|---|---|---|
| 2 | Type | R | 200 | 4.20492 | A | |
| | | C | 200 | 4.17739 | A | |
| | Init | Urand | 200 | 4.28175 | A | |
| | | Xavier | 200 | 4.10057 | | B |
| | $n_L$ | L2 | 200 | 4.21388 | A | |
| | | L4 | 200 | 4.16844 | A | |
| 8 | Type | C | 200 | 2.79881 | A | |
| | | R | 200 | 0.82024 | | B |
| | Init | Xavier | 200 | 2.05504 | A | |
| | | Urand | 200 | 1.56401 | | B |
| | $n_L$ | L2 | 200 | 1.96839 | A | |
| | | L4 | 200 | 1.65066 | | B |

$M$. An argument that could be made is that the difference between the regression and classification types of quantizer are due to the difference in the amount of variables of each network. When $M$ increases, $Q_{NNC}$ will have more variables than $Q_{NNR}$ in the output layer, and because of this difference the DE algorithm favors the quantizer with less variables.

Part of these results are illustrated in Figure 37 which depicts the cumulative errors of the output signals when the error system is fed with $u(k)$ for the cases in which both $Q_{NN}$ are designed with $n_L = \{2,\ 4\}$ and $M = \{2,\ 8\}$ using the Urand initialization method. This error is defined as

$$E(k) := \frac{1}{2}\sum_{i=1}^{k}\left[y_q(i) - y(i)\right]^2 \quad \text{for} \quad k = 1, 2, \ldots, L, \tag{129}$$

which is the considered performance index, Equation (122), over time. The results show that $Q_{NNR}$ and $Q_{NNC}$ have similar performances for $M = 2$ since the

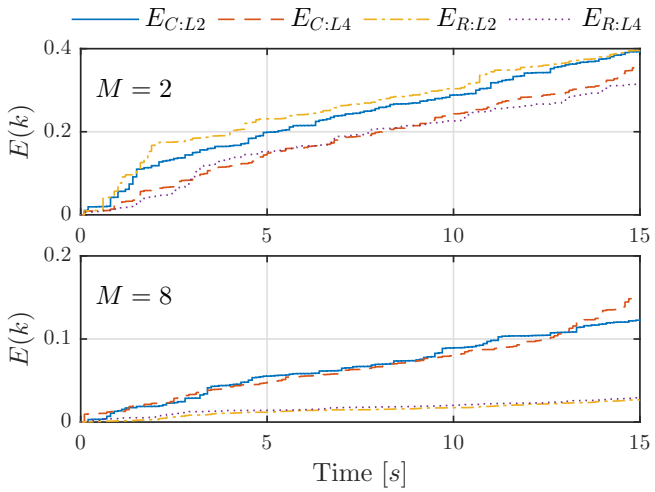



Figure 37: Cumulative errors $E(k)$ when the error system is excited with $u(k)$. The labels of these errors are: $E_{C:L2}$ and $E_{C:L4}$ for $Q_{NNC}$ with $n_L = 2$ and $n_L = 4$, respectively. Similarly, $E_{R:L2}$ and $E_{R:L4}$ are the errors for $Q_{NNR}$.

cumulative errors are close to each other in the considered cases. However, for $M = 8$ the difference is big.

### 5.5.2 Comparison with Linear Dynamic Quantizers

In order to further validate the effectiveness of the neural network quantizer a comparison is carried out with the dynamic quantizer $Q$ previously developed in Sections 2 and 3. This quantizer has a structure similar to the $Q_{NN}$ with the difference that the dynamics is implemented using a linear system instead of a neural network. It is the one represented in Equation (2). The quantizer $Q$ is constructed to operate with linear plants, and its design is carried out using the model of that plant. Thus, since a nonlinear plant is being considered, in this chapter the dynamic quantizer is designed by optimizing an approximation of its performance index given by

$$E_Q(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, d) := \sup_{k \in \{1,2,\ldots,L\}} \operatorname{abs}\left[y_q(k) - y(k)\right], \tag{130}$$

Because the state vector of the considered plant has dimension $n_P = 2$, in

these examples, the matrices $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, and $\boldsymbol{\mathcal{C}}$ adopt the following form

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} 0 & 1 \\ \theta_1 & \theta_2 \end{bmatrix}, \quad \boldsymbol{\mathcal{B}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{\mathcal{C}} = \begin{bmatrix} \theta_3 & \theta_4 \end{bmatrix}. \tag{131}$$

Figure 38 shows the cumulative error, Equation (129), of the output signals when the error system is excited with $u(k)$ for the cases in which $Q_{NNR}$ is designed with $n_L = \{2,\ 4\}$ and Urand initialization method, and $Q$ is designed using the methods described in Section 3 based on DE [106] and covariance matrix adaptation evolution strategy (CMA-ES) [107].

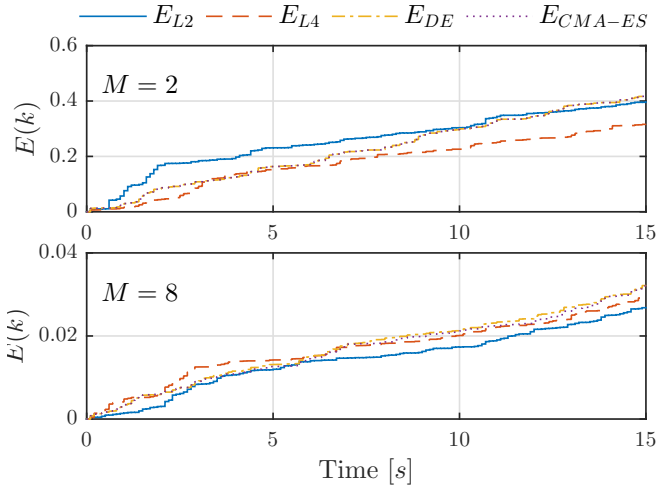



Figure 38: Cumulative errors $E(k)$ resulting when $u(k)$ is applied to the error systems with different $Q_{NN}$ and $Q$. The cumulative error labels are: $E_{L2}$ for $Q_{NN}$ with $n_L = 2$, $E_{L4}$ for $Q_{NN}$ with $n_L = 4$, $E_{DE}$ for $Q$ designed with DE [106], and $E_{CMA-ES}$ for $Q$ designed with CMA-ES [107].

The values of $E(k)$ at the final time shown in the graphics ($k = 15[s]$) are presented in Table 11.

The results indicate that $Q_{NN}$ is superior than $Q$ in dealing with this particular nonlinear plant. This is because the $E(k)$ of $Q_{NN}$ is in each case less than the $E(k)$ of $Q$. Although the difference is small the trend is clear and it can be expected that the difference will grow by applying better training methods to the neural network.

Table 11: Cumulative error $E(k)$ at $k = 150t_S = 15[s]$.

| $M$ | $E_{L2}$ | $E_{L4}$ | $E_{DE}$ | $E_{CMA-ES}$ |
|---|---|---|---|---|
| 2 | 0.396287 | 0.316274 | 0.419084 | 0.419084 |
| 8 | 0.026807 | 0.029498 | 0.032181 | 0.032151 |

### 5.5.3 Activation Functions Comparison

So far only one type of activation function $\mathsf{h}(\cdot)$, the sigmoid function, have been used in the hidden layers to build the neural networks. However, there are other activation functions that can be used. In fact, the search of more effective activation functions is an active area of research [1, 104, 43]. In this section two additional activation functions are considered: the hyperbolic tangent (tanh) and the Rectified Linear Unit (ReLU). These functions were defined, respectively, in Equation (106) and Equation (107) and were shown in Figure 27.

Several numerical simulations were performed to compare the performance of the NNQs built with these functions. The settings of these simulations are the same as in the previous cases where $\mathsf{h} = \text{sigm}$, but they were carried out only for $M = 8$. These simulations were run $N_{run} = 50$ times for each case. The results are summaries in Table 12.

As before, it is difficult to draw conclusions from the table by simple observation. Therefore, the ANOVA test is used to analyse the data. In this case there are four factors that influence the results: $\mathsf{h}$, initialization method, $n_L$ and quantizer type. However, because the influence of $n_L$ is understood the focus in this section will be in the factors: $\mathsf{h}$, initialization method, quantizer type, and the interaction among each other. Therefore, the 3-way ANOVA general linear model of $E(Q_{NN})$ versus quantizer type (Type), initialization method (Init) and activation function $\mathsf{h}$ is considered. The significance level used in this analysis is $\alpha = 0.05$.

The analysis of variance showed that the statistical null hypothesis that all the means are the same was rejected for each single factor and for the combination of them. This means that in each case there is at least one element that significantly differs from the others. The summary of this analysis can be found in Table 20

Table 12: $E(Q_{NN})$ results summary for $\mathsf{h} = \tanh$ and $\mathsf{h} = \text{ReLU}$ ($M = 8$).

| $\mathsf{h}$ | Init. | $n_L$ | Type | Min. | Avg. | Std. Dev. |
|---|---|---|---|---|---|---|
| tanh | Urand | 2 | R | 0.17945 | 0.57343 | 0.73814 |
| | | | C | 0.85707 | 2.17040 | 1.21598 |
| | | 4 | R | 0.24340 | 1.83621 | 1.37026 |
| | | | C | 0.69604 | 1.62132 | 0.68768 |
| | Xavier | 2 | R | 0.17010 | 0.20761 | 0.02103 |
| | | | C | 0.92943 | 1.97048 | 0.78436 |
| | | 4 | R | 0.19868 | 0.25325 | 0.03900 |
| | | | C | 0.66041 | 1.77138 | 0.99609 |
| ReLU | Urand | 2 | R | 0.16532 | 0.75854 | 1.21252 |
| | | | C | 0.70494 | 1.93226 | 0.98668 |
| | | 4 | R | 0.19175 | 3.04153 | 1.69148 |
| | | | C | 0.71718 | 1.93823 | 1.27409 |
| | Xavier | 2 | R | 0.22438 | 2.97202 | 1.24661 |
| | | | C | 0.72398 | 2.12915 | 1.12578 |
| | | 4 | R | 0.22062 | 3.01670 | 1.70763 |
| | | | C | 0.63188 | 2.14319 | 1.03974 |

in Appendix C. After that, the Tukey pairwise comparison is made to see the differences among the quantizer's design elements. The results of this test are summarized in Table 13 for the single factors and some combinations of factors. The rest of the factors combination results are shown in Table 21 in Appendix C.

The results in Table 13 are quite interesting. First, they tell that there is a significant difference between the $Q_{NNR}$ and $Q_{NNC}$, and that $Q_{NNR}$ outperforms $Q_{NNC}$. Second, they show that there is difference between the initialization methods, and that the Urand method exhibits better performance than the Xavier method. These results corroborate the ones obtained in Section 5.5.1 shown in Table 10 for $M = 8$ and $\mathsf{h} = \text{sigm}$. Third, the table shows that the performances of the considered activation functions vary significantly, that the one with the best

Table 13: Tukey pairwise comparison 3-way ANOVA for the activation functions comparison ($M = 8$). Grouping information using the Tukey test and 95% confidence. *Means that do not share a letter are significantly different.*

| Factor | | N | Mean | Grouping | | | |
|---|---|---|---|---|---|---|---|
| Type | C | 600 | 2.23930 | A | | | |
| | R | 600 | 1.32836 | | B | | |
| Init | Xavier | 600 | 1.89033 | A | | | |
| | Urand | 600 | 1.67733 | | B | | |
| h | ReLU | 400 | 2.24145 | A | | | |
| | sigm | 400 | 1.80953 | | B | | |
| | tanh | 400 | 1.30051 | | | C | |
| Type*Init | C Xavier | 300 | 2.24700 | A | | | |
| | C Urand | 300 | 2.23161 | A | | | |
| | R Xavier | 300 | 1.53365 | | B | | |
| | R Urand | 300 | 1.12306 | | | C | |
| Type*h | C sigm | 200 | 2.79881 | A | | | |
| | R ReLU | 200 | 2.44720 | | B | | |
| | C ReLU | 200 | 2.03571 | | | C | |
| | C tanh | 200 | 1.88340 | | | C | |
| | R sigm | 200 | 0.82024 | | | | D |
| | R tanh | 200 | 0.71763 | | | | D |

performance is h = tanh, and that the one with lowest performance is h = ReLU.

These results, however, may vary depending of specific cases. Then, it is interesting to analyze the performance of the combination of factors. Hence, the Type∗Init combination tells that for $Q_{NNC}$ there is no significant difference between the initialization methods, but that for $Q_{NNR}$ there is, and that the Urand method gives the best results in this case. The Type∗h combination shows that for $Q_{NNC}$ the activation functions with the best performance are ReLU and tanh with no significant difference between them. On the other hand, for $Q_{NNR}$ the activation functions with the best performance are sigm and tanh also with

no significant difference between them. Finally, Figure 39 and Figure 40 show some plots that help to clarify the influence of these factors and how they interact with each other.

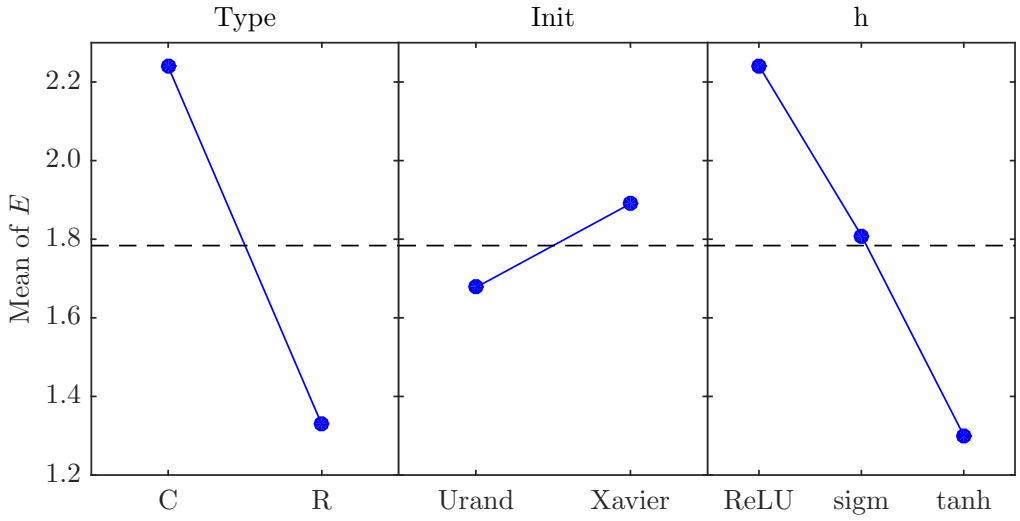



Figure 39: 3-way ANOVA main effects plot for $E(Q_{NN})$ fitted means.

## 5.6 NNQ for Multiple Input Signals

So far in this chapter the design of the NNQs have been carried out only for one reference signal $u(k)$. This can pose severe limitations to the system. Ideally, a NNQ should work with any type of input signal. Thus, the quantizer should learn about the plant's internal structure not only its response to one specific signal. This can be done, perhaps, using system identification techniques and theory. However, the approach considered in this chapter is to feed the system with a variety of signals with different frequencies. The assumption that the signals are bounded in the interval $U = [-1, 1]$ is maintained.

The performance of the NNQ in this case is evaluated using the same error system as in the case for a single input signal given in Figure 31 . Also, the performance index is a generalization of the one described in Section 5.3.3, Equation (122), for a single input signal. Thus, the performance index of NNQ

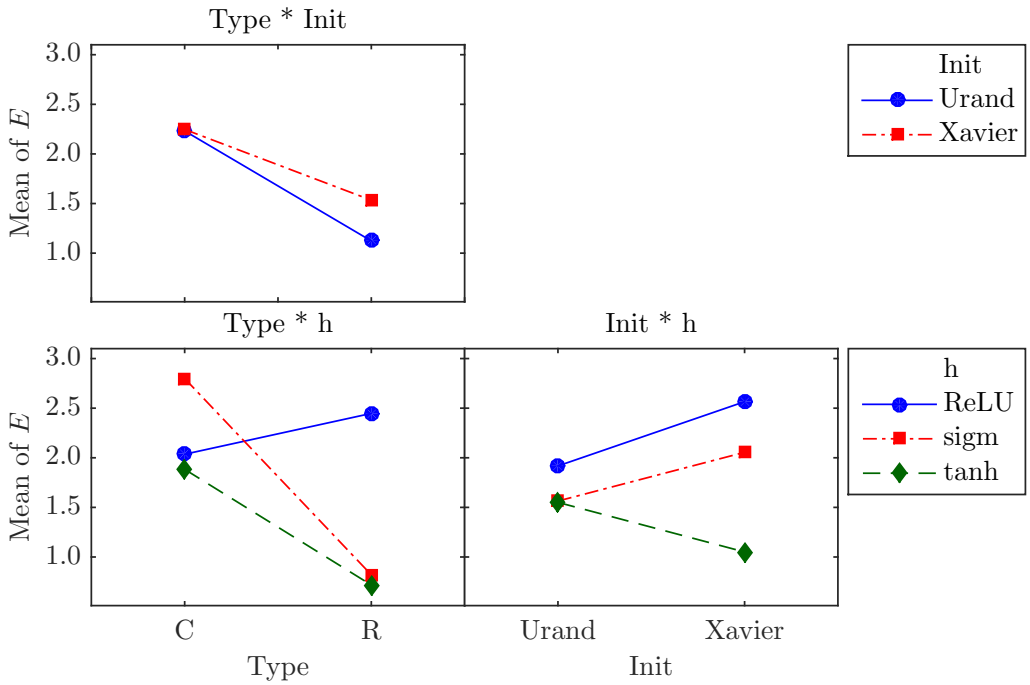



Figure 40: 3-way ANOVA interaction plot for $E(Q_{NN})$ fitted means.

for multiple $u(k)$ is defined as follows

$$E_m(\mathbf{w}, d) = \sum_{i=1}^{n_u} E(u_i(k), \mathbf{w}, d) = \frac{1}{2} \sum_{i=1}^{n_u} \sum_{k=0}^{n_s} [y_q(u_i(k), \mathbf{w}, d) - y_i(k)]^2 \qquad (132)$$

where, $n_u$ represents the considered amount of input signals. This number is arbitrary, however, the more signals are used the bigger the neural network should be. Notice, that it is assumed that there is no interaction among the signals, they are independent from each other. The design problem is the same as in the case for a single input signal which is described in Section 5.3.4 with the difference that the performance index is the one in Equation (132). This design problem is solved again using differential evolution. The main disadvantage of this design method is that the time used for the training can be high. It takes $n_u$ times longer to train this quantizer than in the case with a single $u(k)$.

In order to verify the effectiveness of the NNQs when they are designed for multiple $u(k)$ various numerical simulations were performed. The considered plant is the one given in Equation (127), the sampling time is $t_S = 0.1$, and the

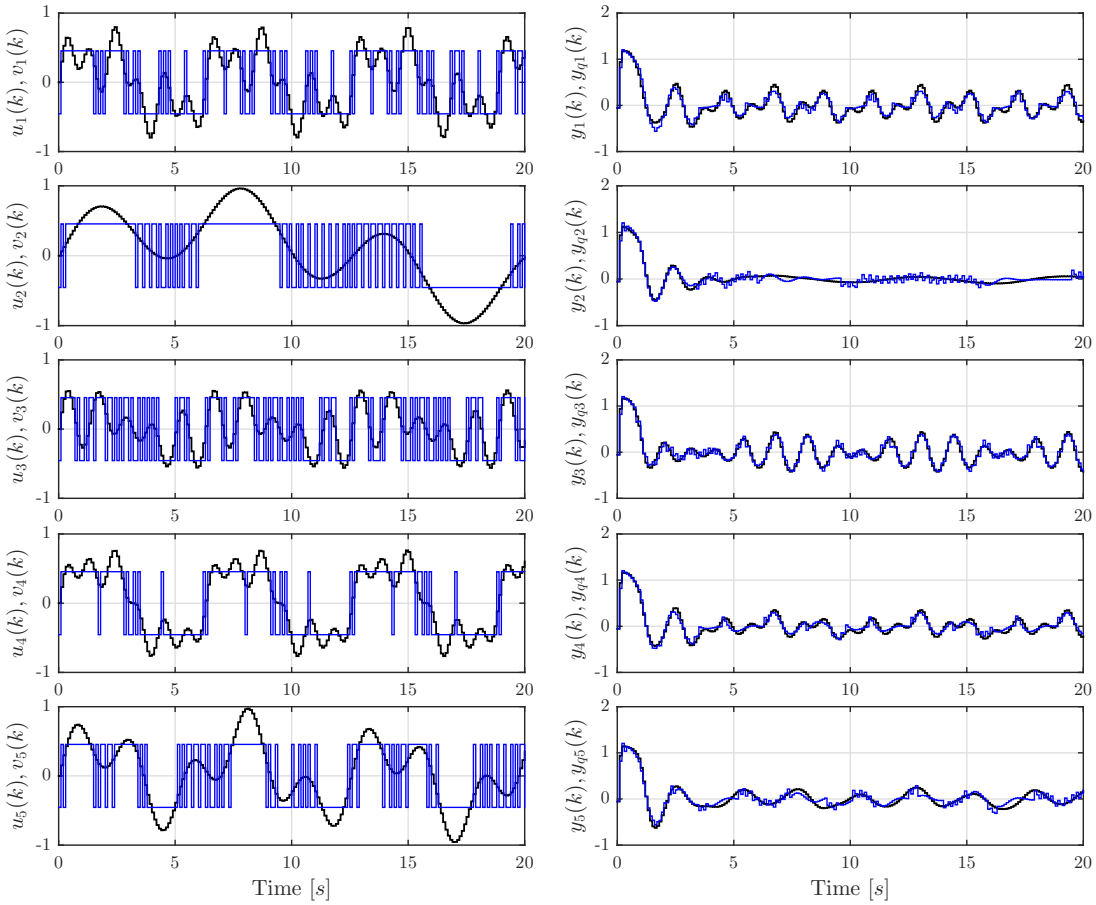
evaluation interval is $L = n_s = 500$. The input signals used in these simulations are the same the ones used in Section 4.5 from Equation 98 to Equation 102.

The simulations in this case were carried out for the regression type of quantizer $Q_{NNR}$ with $M = 2$ and a neural network given by $\boldsymbol{K} = [10,\ 10,\ 10,\ 1]$ ($n_L = 4$, $n_{\mathsf{w}} = 352$). The configuration of the DE parameters are the same as in the numerical examples in Section 5.5 and the simulations are run $N_{run} = 10$ times. After the simulations ended and to verify that the designed quantizer works properly the error system was fed with the considered $u(k)$s. The resulting signals are shown in Firgure 41. This figure shows the form of the considered input signals $u_i(k)$ $(i = 1, 2, \ldots, 5)$ in black in the left column. The signals in blue are the respective quantized signals $v_i(k)$ $(i = 1, 2, \ldots, 5)$ that go to the plant, they have only two quantization levels. The right column depicts the outputs, the ideal ones $y_i(k)$ $(i = 1, 2, \ldots, 5)$ in black and the ones due to quantization in blue $y_{q,i}(k)$ $(i = 1, 2, \ldots, 5)$. It is possible to see that the error between the output signals is small and how the quantized outputs follow the ideal ones mostly well. The biggest errors are produced by $u_2(k)$ and $u_5(k)$ where the frequency is low and the amplitude for their respective $y(k)$ is small. Therefore, it is possible to say that the NNQ for multiple $u(k)$s works well in a certain range of frequencies. This range of frequencies is probably a function of the sampling time and the plant.

To further verify that the NNQ for multiple $u(k)$s works properly, the same quantizer designed previously for a single input signal, $u_1(k)$, is fed with the other considered input signals . The results are shown in Figure 42. This figure clearly shows the advantage of designing the quantizer for multiple $u(k)$s.

### 5.6.1 Noise Addition Effect

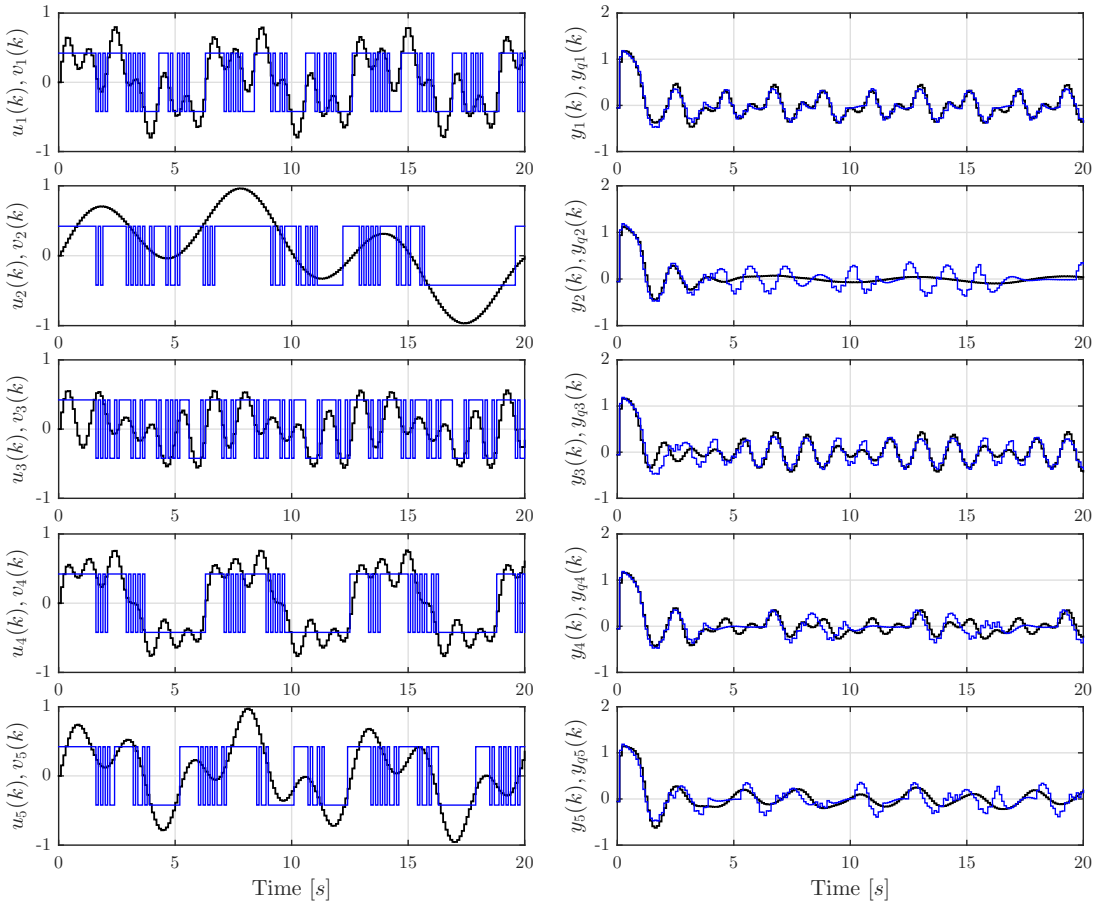
In all the numerical examples previously developed some degree of noise was always added to the input signals to perform the training. The idea is to make the quantizer robust to the presence of noise. The noise added in this examples is white noise with amplitude 2.5% of the input signals semi range ($0.025(u_{max} - u_{min})/2$), and it is added ones for each run of the DE algorithm. This means that the $u(k)$s used to sample the outputs are noisy, and that the noise does not change during a run of DE.




(a) Inputs. (b) Outputs.

Figure 41: Signals resulting from exciting the designed $Q_{NNR}$ with the set of considered $u(k)$. The signals in black are the signals without quantization ($u(k)$, $y(k)$) and the ones in blue are the signals when quantization is applied ($v(k)$, $y_q(k)$). Each row shows the inputs an outputs for one particular $u(k)$.

This section studies the effects of additive noise in the design of NNQs. For this purpose three cases are considered. First, no noise case, the quantizer is trained without adding noise. Second, fixed noise case, the noise is added to the input signal only at the beginning each run of DE. This is the case used so far. Third, variable noise case, the noise is added to the input signal in each evaluation of the cost function in the DE algorithm. The fixed noise case has the




(a) Inputs.

(b) Outputs.

Figure 42: Signals resulting from feeding the $Q_{NNR}$ designed for the single input signal $u_1(k)$ with the set of signals given from Equation (98) to Equation (102). The signals in black are the signals without quantization ($u(k)$, $y(k)$) and the ones in blue are the signals when quantization is applied ($v(k)$, $y_q(k)$). Each row shows the inputs an outputs for one particular $u(k)$.

advantage that the simulations are relatively fast in comparison with the variable noise case.

Thus, simulations are performed for the cases with no noise and variable noise. The settings of these simulations are the same as in Section 5.6, and the noise added in the case of variable noise is again 2.5% of the input signals semi range.

The simulations are run $N_{run} = 10$ times for each case. The results of these simulations are summarized in Table 14. They are the ones in bold faced letters. In this table the results for the case with fixed noise are added as well. Obviously, the case with no noise gives the smallest performance index.

Table 14: $E(Q_{NN})$ analysis of NNQs for multiple signals with the addition of white noise ($N_{run} = 10$).

| Noise % | Noise Type | Min. | Avg. | Std. Dev. |
|---|---|---|---|---|
| 0% | **none** | **10.97863** | **11.54412** | **0.47002** |
| | fixed | 12.18897 | 12.97703 | 0.59440 |
| | variable | 12.18248 | 13.21686 | 0.65075 |
| 2.5% | none | 12.44563 | 13.57155 | 0.63996 |
| | **fixed** | **12.71939** | **13.54894** | **0.55361** |
| | **variable** | **12.35034** | **13.56092** | **0.74476** |
| 5% | none | 13.79437 | 15.14641 | 0.85155 |
| | fixed | 14.00222 | 14.95164 | 0.53057 |
| | variable | 13.90767 | 14.74668 | 0.54413 |
| 10% | none | 15.75112 | 17.24837 | 1.02287 |
| | fixed | 16.48679 | 17.69108 | 0.74287 |
| | variable | 16.82818 | 18.41722 | 1.68169 |
| Other $u(k)$s | none | 18.78743 | 19.99981 | 1.10599 |
| | fixed | 18.48116 | 20.98486 | 1.15570 |
| | variable | 19.32231 | 21.11520 | 1.02833 |

The rest of the values in Table 14 are the summary of the performances indexes obtained by feeding all the quantizers designed for 2.5% of noise with the same input signals perturbed with different percentages of noise. This is done to test the robustness of the designed quantizers to the presence of white noise and to get an idea of the best way to design these quantizers. Furthermore, in the last category 'Other $u(k)$s', the quantizers instead of being perturbed with white noise, were excited with a set of signals different than the ones for which the quantizers were designed for.

From these results the one way ANOVA test for $\alpha = 0.05$ is carried out for each level of noise being the factor the considered noise addition cases. This analysis showed that the only instance where the are significant differences is the case for 0% of noise. Then, the Fisher LSD and Tukey's test were run for this case showing that the case with no noise gives the smallest mean. However, this result is trivial since the quantizer was designed for that purpose. Thus, although from the values of Table 14 it may seem that the case with no noise is better than the others, there were no significant differences to be found. In conclusion, the addition of noise in the training of NNQs seems to be not necessary.

### 5.6.2 Performance Indexes Comparison

In this section the effects of using performance indexes different than the one described in Section 5.3.3 are explored. This performance index is based on the sum of square errors and it is also the one used for the NNQ trained for multiple signals, Equation (132). In addition to this performance index there are some others that can be used. For example, the sum of the absolute values of the errors and the supremum of the errors' absolute values. These performance indexes are represented in Equation (134) and Equation (135), respectively. The one in Equation (133) is the sum of square errors, that it is shown here again just for comparison.

$$\text{Es2:} \quad E(u_i(k), \mathbf{w}, d) = \sum_{k=0}^{n_s} \left[y_q(u_i(k), \mathbf{w}, d) - y(k)\right]^2, \tag{133}$$

$$\text{Esa:} \quad E(u_i(k), \mathbf{w}, d) = \sum_{k=0}^{n_s} \operatorname{abs}\left(y_q(u_i(k), \mathbf{w}, d) - y(k)\right), \tag{134}$$

$$\text{Ema:} \quad E(u_i(k), \mathbf{w}, d) = \max_{\substack{u \in U \\ k \in \{0,1,2,\ldots,n_s\}}} \operatorname{abs}\left(y_q(u_i(k), \mathbf{w}, d) - y(k)\right), \tag{135}$$

for $i = 1, 2, \ldots, n_u$. For better reference the identifiers Es2, Esa and Ema are assigned to each performance index. These indexes can be used to designed the NNQs for a single input signal or they can be inserted into Equation (132), replacing the sum of squares errors, to design NNQs for multiple input signals.

To verify the effectiveness of these performance indexes and to compare them with the one previously used, Es2, several numerical examples were carried out.

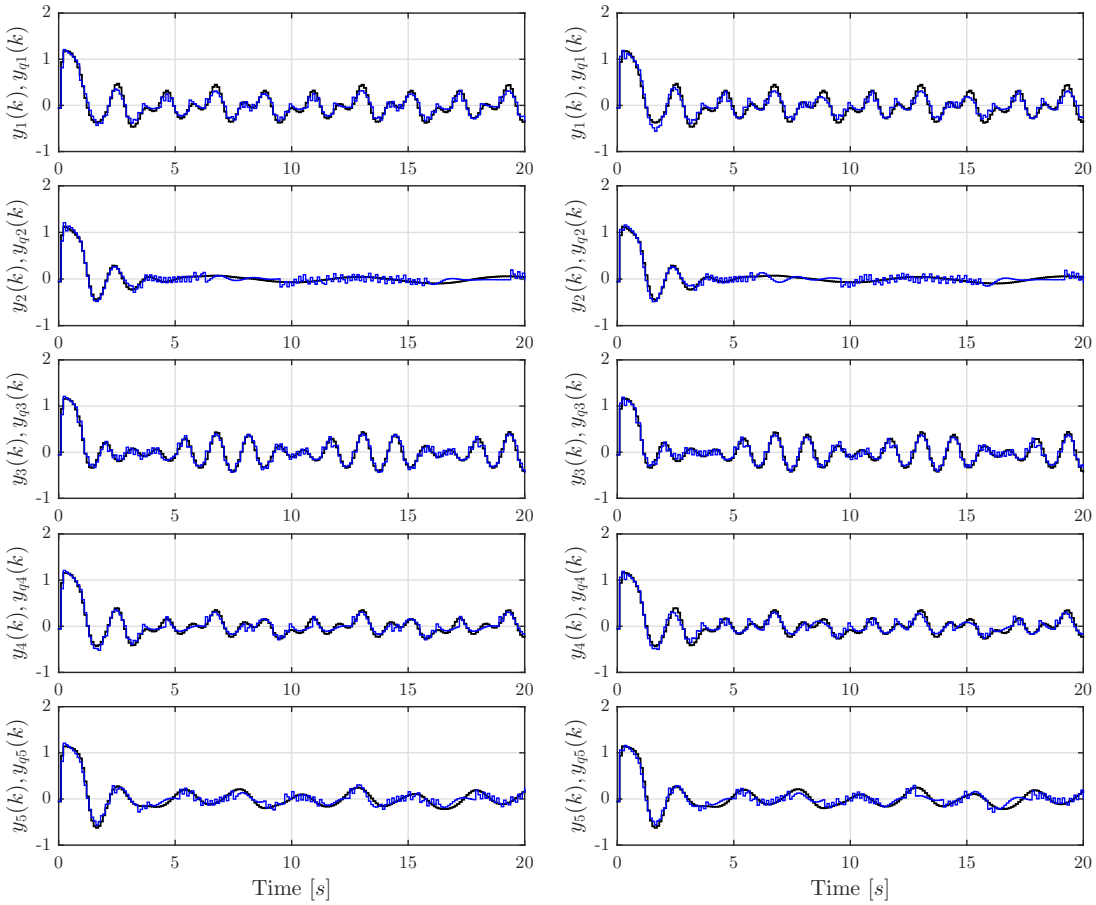
The settings of these examples are the same as in Section 5.6 with the exception, of course, of the performance index used. Thus, for Esa and Ema the simulations are run $N_{run} = 10$ times. After the simulations ended, to verify the effectiveness of the the designed quantizers, the error system is fed with the considered input signals. The quantizers work properly, as it is shown in Figure 43. These output signals can be compared with their equivalents in Figure 41b obtained from the system with the quantizer designed for Es2.

Now the problem is how to compare these different performance indexes. Usually, these indexes are the ones that determine the system performance but in this case the performance of these indexes is being compared. Therefore, the comparison is carried out as follows. Each quantizer designed for a specific performance index is fed with the considered input signals and the value of the other performance indexes are calculated based on the error signal. After that, a statistical analysis is run for the values of each index to see if there is significant difference in each case. The summary of the resulting values is shown in Table 15.

Table 15: $E(Q_{NN})$ analysis of NNQs for multiple signals designed with different expressions of $E(Q_{NN})$ ($N_{run} = 10$).

| Evaluated with | Designed with | Min. | Avg. | Std. Dev. |
|---|---|---|---|---|
| Es2 | Es2 | 10.97863 | 11.54412 | 0.47002 |
| | Esa | 11.03669 | 12.13267 | 0.77098 |
| | Ema | 13.51830 | 16.46565 | 2.12234 |
| Esa | Es2 | 133.75732 | 137.56370 | 2.20182 |
| | Esa | 133.15346 | 138.12278 | 3.28806 |
| | Ema | 148.04155 | 164.15430 | 9.80238 |
| Ema | Es2 | 0.20996 | 0.22437 | 0.00781 |
| | Esa | 0.20278 | 0.22691 | 0.01148 |
| | Ema | 0.19847 | 0.21099 | 0.01070 |

Then, for each case of 'Evaluated with' in Table 15 a one way ANOVA test with significance level $\alpha = 0.05$ is performed. In each considered case there was significant difference among the performance indexes. Then, to see which index




(a) Esa (b) Ema

Figure 43: Output signals comparison for NNQs trained for multiple $u(k)$s and different performance indexes. The signals in black represent the ideal outputs $y(k)$ and the ones in blue are the outputs when quantization is applied $y_q(k)$. Each row shows the inputs an outputs for one particular $u(k)$.

has better performance the Fisher LSD and Tukey's tests were carried out. Both tests gave similar results. The summary of the Tukey's tests is shown Table 16. These results evidence the following relations. First, that there is no significant difference between the Es2 and Esa indexes, and that in each case Es2 has a smaller mean. Thus, although there is no significant difference, it seems that Es2 has slightly better performance than Esa. Second, that Ema is significantly

Table 16: Tukey pairwise comparison 1-way ANOVA for performance indexes comparison. Grouping Information Using the Tukey Method and 95% confidence. *Means that do not share a letter are significantly different.*

| Evaluated with | Factor | N | Mean | Grouping | |
|---|---|---|---|---|---|
| Es2 | Ema | 10 | 16.466 | A | |
| | Esa | 10 | 12.133 | | B |
| | Es2 | 10 | 11.544 | | B |
| Esa | Ema | 10 | 164.15 | A | |
| | Esa | 10 | 138.12 | | B |
| | Es2 | 10 | 137.56 | | B |
| Ema | Esa | 10 | 0.22691 | A | |
| | Es2 | 10 | 0.22437 | A | |
| | Ema | 10 | 0.21099 | | B |

different than the other indexes and that only for the case of Ema it shows the best performance. Accordingly, it is safe to say the performance indexes Es2 and Esa are preferred for the design of NNQs.

## 5.7 Summary

This chapter introduces the concept of neural network quantizers (NNQs) that are designed using a set of the inputs and outputs of the plant. These quantizers are aimed to systems in which the model of the plant is unknown or unreliable. They are constructed using feedforward neural networks and static quantizers. Two types of NNQs are proposed: regression and classification types. In addition, a design method based on differential evolution is proposed for these quantizers.

By means of several numerical examples it was found that both types of NNQs are effective along side with their DE based design method. Furthermore, many variations were considered in the construction of these quantizers. These variations are reflected in the number of quantization levels ($M = \{2, 8\}$), in the number of layers of the network ($n_L = \{2, 4\}$), in the type of network initial-

ization technique (Urand, Xavier) and in the hidden layers' activation functions (sigm, tanh, ReLU). Several conclusions were reached based on the analysis of variance performed on the simulations results. Some of the most important are that the quantizers based on regression outperform the ones based on classification, that the best initialization method is the random uniform (Urand), and that the activation function that gives the best performance is tanh.

The design of NNQs for multiple input signals was also considered. The results show that these quantizers are effective. Moreover, the effects of additive white noise in the design of NNQs was studied. The results shows that the is no significant difference in adding noise or not when training the quantizer. Finally, impact of using different types of performance indexes in the quantizer's design was explored. All the indexes gave good results, and one based on the sum of square errors seems to perform the best.

There are, however, still many issues that need to be addressed. Some drawbacks of theses quantizers are the long time required for the design and that the precision decreases when the number of input signals increases. Some future works include the implementation of the pre-training of the neural network, and the used of metaheuristics adapted to high dimensional optimization.

# 6. Conclusion

## 6.1 Summary

In this thesis several types of noise-shaping quantizers for networked control systems (NCSs) were proposed. Each of these quantizers is designed to solve certain problems that affect the NCSs. First, a finite-level dynamic quantizer design method based on metaheuristics is proposed. The metaheuristics considered in this study are CMA-ES, DE and FA. Second, a switching type dynamic quantizer known as event-triggered quantizer is proposed. This quantizer is actuated by an event-generator based on Gaussian functions that is attached to the plant and sends the data only when needed. Finally, the neural network quantizer is introduced. This quantizer is implemented with neural networks and a time series of the plant's inputs/outputs. This quantizer does not need a model of the plant, and could be applied to any type of system. Several numerical simulations were carried out to verify the effectiveness of these quantizers and their proposed design methods. From the results of theses simulations several conclusions were made.

For the design of finite-level dynamic quantizers subjected to data rate constraints it was found that the CMA-ES and DE based design methods show in general very good performance in terms of success rate and convergence time. They both have advantages and disadvantages comparing to each other. The method based on CMA-ES shows a slightly better performance than DE, specially when considering a relatively small amount of individuals and number of generations, besides the execution time is smaller. Moreover, CMA-ES carries the big advantage that it does not require the tuning of any control parameter. Their values are already optimized and established by default. Nevertheless, it is worth to mention that DE is very easy to implement contrary to CMA-ES. Compared to the other metaheuristic based design methods, it was verified that the performance of the CMA-ES and DE based methods are quite better than the ones based on PSO and FA. For these reasons, it is possible to say that the methods based on CMA-ES and DE are very reliable for the design of the finite-level dynamic quantizer, but the ones based on FA and PSO are not.

In regard to the proposed event-triggered dynamic quantizer it was found

that it works well along with its DE based design methods. It is important to remember that the reduction of the traffic in the network will negatively affect the performance of the system by increasing outputs errors. Thus, the designer should very carefully choose the maximum NUR value having in account the network capabilities and the system limitations. From the considered design methods, the ones in which the dynamic quantizer is designed along side with the event-generator show superior performance. Nevertheless, the methods in which dynamic quantizer and the event-generator are designed separately are useful when the dynamic quantizer is already implemented in the system. The version VV, where the plant retains the previous data until a new one arrives, is superior to the version V0, where the plant assumes zero when no data arrives. The version V0 is only better for the case of $M = 2$, but its performance decreases for $M > 2$. Moreover, the design methods in which the Gaussian functions' means and variances are variables generate less errors than the methods in which they are fixed. However, for these methods it is important to consider meta heuristics adapted for high dimensional optimization.

The proposed neural network quantizers showed good performance as well as their DE based design method. These quantizers are built with a neural network and a static quantizer. The statistical analysis performed on the simulations results showed that from the two types of proposed NNQs the ones based on regression $Q_{NNR}$ outperform the ones based on classification $Q_{NNC}$. Additionally, two initialization methods were considered, random uniform (Urand) and Xavier, being the first one the most effective. From the considered hidden layers' activation functions (sigm, tanh, ReLU) the one that gives the quantizers with the best performance is tanh. These results vary slightly when considering $Q_{NNR}$ and $Q_{NNC}$ separately. For $Q_{NNR}$ the best initialization method is Urand and the best activation functions are tanh and sigm with no significant difference. On the other hand, for $Q_{NNC}$ there is no significant difference between the considered initialization methods and the activation functions that give the best performances are tanh and ReLU again with no significant difference. In addition, it was found that the NNQs outperform the finite-level dynamic quantizers considered before, when dealing with plants with unknown structure.

## 6.2 Limitations

Although the proposed quantizers partially solve the considered problems there are subjected to some limitations. The first one is that the ETQs and NNQs are designed using a known input signal. This is a very restrictive condition. To overcome this constraint it is important to find theoretical expressions for the performance indexes that are independent of the input signals. However, this expression may not exist. Therefore, in this study the design of quantizers for a set of input signals is also considered. In the case of NNQs it is expected that using a big enough set of signals the neural network will capture information of the plant's internal structure and the quantizer will be able to work properly independently of the input signal.

The second limitation of this study is the time that it takes to design the quantizers. In particular, the design of ETQs and NNQs for multiple signals take quite a lot of time. It is important to find ways to reduce the quantizer's design time. For instance, some direct ways to optimize the design time include to parallelize the metaheuristics execution, to use powerful computers and to codify the quantizer's design program using compiled programming languages like C/C++.

Finally, the metaheuristics used in this study are not well suited to perform the optimization of high dimensional problems ($n > 100$). In order to improve the quantizers' performance it is necessary to use metaheuristics designed for large scale global optimization problems [71]. Examples of these metaheuristics include variations of DE and CMA-ES or the use of metaheuristic based on cooperative coevolution [98, 13].

## 6.3 Possible Research Directions

This thesis is just the starting point in the development of the proposed noise-shaping quantizers. There are many issues that need to be addressed regarding to their structure and design. The performance of the proposed quantizers can be greatly improved. For instance, it is still needed to find an analytical expression for the optimal finite-level dynamic quantizer subjected to data rate constraints. Besides, the event-triggered quantization offers a fruitful field for research. Many

types of event-triggered quantizer can be constructed and the event-triggered and self-triggered control theories should be applied on their analysis and design. In regard to the neural network quantizers there is a lot of work to do. For example, the use of metaheuristics designed for large scale global optimization, the search for techniques to accelerate the design time of these quantizers, and the use of strategies from the system identification field in the design of NNQs are future research directions. Finally, other problems that affect the NCSs should be considered as well in the design of theses quantizers. These problems include package looses and delays in the delivery of data, the effects of the noise in the channel, and the security of the control system against external attacks.

# Acknowledgements

I would like to express my sincere gratitude to Professor Kenji Sugimoto and Associate Professor Takamitsu Matsubara for accepting me into their laboratory as a student, for their kindness, thoughtful advice and assistance throughout my study. To the Assistant Professors Masaki Ogura, Taisuke Kobayashi and Cui Yunduan for their kind advice and helpful comments. I would also like to thank Professor Shoji Kasahara for his contribution on the thesis committee and to Associate Professor H. Okajima, Kumamoto University, for his valuable comments. To Hideko Hayashi for helping me out with all the paper work required during my courses. I also would like to thank my parents Gerardo and Norma, my aunt Sofia and my siblings, Mariano, Belen and little Gerardo, who have been supporting me from the distance and for all their prayers. To my girlfriend Desislava for taking care of me and supporting me in the difficult times. I would also like to thank the Government of Japan through its scholarship program Monbukagakusho for granting me the financial support that enabled me to come to Japan for my education and research. Finally, I wish to specially thank Associate Professor Yuki Minami, Osaka University, for guiding me through the master and doctoral courses, for his wise advice, kindness and patient, for encouraging me to surpass my limits and for being a true friend.

# References


[1] F. Agostinelli, M. D. Hoffman, P. J. Sadowski, and P. Baldi. Learning activation functions to improve deep neural networks. *CoRR*, abs/1412.6830, 2014.

[2] E. Alpaydin. *Introduction to Machine Learning*, chapter Multilayer Perceptrons, pages 233–277. The MIT Press, 2nd edition, 2010.

[3] P. J. Antsaklis and J. Baillieul. Special issue on technology of networked control systems. *Proceedings of the IEEE*, 95(1):5–8, 2007.

[4] S. Azuma and T. Sugie. Optimal dynamic quantizers for discrete-valued input control. *Automatica*, 44(2):396–406, 2008.

[5] S. Azuma and T. Sugie. Synthesis of optimal dynamic quantizers for discrete-valued input control. *IEEE Transactions on Automatic Control*, 53(9):2064–2075, 2008.

[6] S. Azuma and T. Sugie. Dynamic quantization of nonlinear control systems. *IEEE Transactions on Automatic Control*, 57(4):875–888, 2012.

[7] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[8] L. Bianchi, M. Dorigo, L. Gambardella, and W. J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287, 2009.

[9] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[10] I. Boussaïd, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013. Prediction, Control and Diagnosis using Advanced Neural Computations.

[11] R. W. Brockett and D. Liberzon. Quantized feedback stabilization of linear systems. *IEEE Transactions on Automatic Control*, 45(7):1279–1289, 2000.

[12] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009.

[13] R. Chandra, M. Frean, and M. Zhang. On the issue of separability for problem decomposition in cooperative neuro-evolution. *Neurocomputing*, 87:33–40, 2012.

[14] C.-K. Chang, J. M. Overhage, and J. Huang. An application of sensor networks for syndromic surveillance. In *Proceedings of the 2005 IEEE Networking, Sensing and Control*, pages 191–196, 2005.

[15] C.-T. Chen. *Linear System Theory and Design*. Oxford University Press, Inc., New York, NY, USA, 3rd edition, 1998.

[16] M.-Y. Chow and Y. Tipsuwan. Network-based control systems: a tutorial. In *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society*, volume 3, pages 1593–1602, 2001.

[17] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[18] J. de Jesús Rubio. Discrete time control based in neural networks for pendulums. *Applied Soft Computing*, 68:821–832, 2018.

[19] C. De Persis. Robust stabilization of nonlinear systems by quantized and ternary control. *Systems & Control Letters*, 58(8):602–608, 2009.

[20] C. De Persis and A. Isidori. Stabilizability by state feedback implies stabilizability by encoded state feedback. *Systems & Control Letters*, 53(3):249–258, 2004.

[21] C. C. de Wit, F. R. Rubio, J. Fornes, and F. Gomez-Estern. Differential coding in networked controlled linear systems. In *Proceedings of the 2006 American Control Conference*, pages 4177–4182, 2006.

[22] J.-C. Delvenne. An optimal quantized feedback strategy for scalar linear systems. *IEEE Transactions on Automatic Control*, 51(2):298–303, 2006.

[23] N. Dong and Z. Chen. A novel data based control method based upon neural network and simultaneous perturbation stochastic approximation. *Nonlinear Dynamics*, 67(2):957–963, 2012.

[24] N. Elia and S. K. Mitter. Stabilization of linear systems with limited information. *IEEE Transactions on Automatic Control*, 46(9):1384–1400, 2001.

[25] F. Fagnani and S. Zampieri. Stability analysis and synthesis for scalar linear systems with a quantized feedback. *IEEE Transactions on Automatic Control*, 48(9):1569–1584, 2003.

[26] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 7th edition, 2014.

[27] M. Fu and L. Xie. The sector bound approach to quantized feedback control. *IEEE Transactions on Automatic Control*, 50(11):1698–1711, 2005.

[28] M. Fu and L. Xie. Finite-level quantized feedback control for linear systems. *IEEE Transactions on Automatic Control*, 54(5):1165–1170, 2009.

[29] H. Gao and T. Chen. A new approach to quantized feedback control systems. *Automatica*, 44(2):534–542, 2008.

[30] E. Garcia and P. J. Antsaklis. Model-based event-triggered control for systems with quantization and time-varying network delays. *IEEE Transactions on Automatic Control*, 58(2):422–434, 2013.

[31] X. Ge, F. Yang, and Q.-L. Han. Distributed networked control systems: A brief overview. *Information Sciences*, 380:117–131, 2017.

[32] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256, 2010.

[33] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[34] G. C. Goodwin, K. Lau, and M. G. Cea. Control with communication constraints. In *Proceedings of the 12th International Conference on Control Automation Robotics Vision*, pages 1–10, 2012.

[35] G. C. Goodwin, E. I. Silva, and D. E. Quevedo. A brief introduction to the analysis and design of networked control systems. In *Proceedings of the 2008 Chinese Control and Decision Conference*, pages 1–13, 2008.

[36] R. A. Gupta and M. Chow. Networked control system: Overview and research trends. *IEEE Transactions on Industrial Electronics*, 57(7):2527–2535, 2010.

[37] N. Hansen. The CMA evolution strategy: a comparing review. In J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[38] N. Hansen. Benchmarking a bi-population cma-es on the bbob-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2389–2396, 2009.

[39] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[40] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[41] T. Hayakawa, H. Ishii, and K. Tsumura. Adaptive quantized control for linear uncertain discrete-time systems. *Automatica*, 45(3):692–700, 2009.

[42] T. Hayakawa, H. Ishii, and K. Tsumura. Adaptive quantized control for nonlinear uncertain systems. *Systems & Control Letters*, 58(9):625–632, 2009.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of*

*the 2015 IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[44] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. In *Proceedings of the IEEE 51st Annual Conference on Decision and Control*, pages 3270–3285, 2012.

[45] J. Hespanha, M. Mclaughlin, G. S. Sukhatme, M. Akbarian, R. Garg, and W. Zhu. Haptic collaboration over the internet. In *Proceedings of the 5th Phantom Users Group Workshop*, 2000.

[46] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, 2007.

[47] K. Hikichi, H. Morino, I. Arimoto, K. Sezaki, and Y. Yasuda. The evaluation of delay jitter for haptics collaboration over the internet. In *Proceedings of the 2002 IEEE Global Telecommunications Conference*, volume 2, pages 1492–1496, 2002.

[48] W. Holderbaum. Neural network application to linear systems with binary inputs. In *Proceedings of the 42nd IEEE International Conference on Decision and Control*, volume 5, pages 5103–5108, 2003.

[49] W. Holderbaum. Application of neural network to hybrid systems with binary inputs. *IEEE Transactions on Neural Networks*, 18(4):1254–1261, 2007.

[50] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[51] S. Hu and D. Yue. Event-triggered control design of linear networked systems with quantizations. *ISA Transactions*, 51(1):153–162, 2012.

[52] J. Ilonen, J.-K. Kamarainen, and J. Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1):93–105, 2003.

[53] H. Inose, Y. Yasuda, and J. Murakami. A telemetering system by code modulation-$\Delta$-$\Sigma$ modulation. *IRE Transactions on Space Electronics and Telemetry*, SET-8(3):204–209, 1962.

[54] H. Ishii and T. Basar. An analysis on quantization effects in $H^\infty$ parameter identification. In *Proceedings of the 2004 IEEE International Conference on Control Applications*, pages 468–473, 2004.

[55] H. Ishii and B. A. Francis. Quadratic stabilization of sampled-data systems with quantization. *IFAC Proceedings Volumes*, 35(1):67–72, 2002. 15th IFAC World Congress.

[56] J. Jaglin, C. C. de Wit, and C. Siclet. Delta modulation for multivariable centralized linear networked controlled systems. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 4910–4915, 2008.

[57] R. Javed, G. Mustafa, A. Q. Khan, and M. Abid. Networked control of a power system: A non-uniform sampling approach. *Electric Power Systems Research*, 161:224–235, 2018.

[58] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[59] A. Kheirkhah, D. Aschenbrenner, M. Fritscher, F. Sittner, and K. Schilling. Networked control systems with application in the industrial tele-robotics. *IFAC-PapersOnLine*, 48(10):147–152, 2015. 2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics CESCIT 2015.

[60] E. Konaka. Model-free controller design for discrete-valued input systems based on autoencoder. In *Proceedings of the 2016 SICE Annual Conference*, pages 685–690, 2016.

[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[62] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 507–514, 2012.

[63] K. Li and J. Baillieul. Robust quantization and coding for multidimensional linear systems under data rate constraints. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 2, pages 1920–1925, 2004.

[64] K. Li and J. Baillieul. Robust quantization for digital finite communication bandwidth (dfcb) control. *IEEE Transactions on Automatic Control*, 49(9):1573–1584, 2004.

[65] D. Liberzon. Hybrid feedback stabilization of systems with quantized signals. *Automatica*, 39(9):1543–1554, 2003.

[66] D. Liberzon. On stabilization of linear systems with limited information. *IEEE Transactions on Automatic Control*, 48(2):304–307, 2003.

[67] D. Liberzon and D. Nesic. Input-to-state stabilization of linear systems with quantized state measurements. *IEEE Transactions on Automatic Control*, 52(5):767–781, 2007.

[68] J. Liu and N. Elia. Quantized feedback stabilization of non-linear affine systems. *International Journal of Control*, 77(3):239–249, 2004.

[69] Q. Liu, Z. Wang, X. He, and D. Zhou. A survey of event-based strategies on control and estimation. *Systems Science & Control Engineering*, 2(1):90–97, 2014.

[70] T. Liu, Z. Jiang, and D. J. Hill. Quantized stabilization of strict-feedback nonlinear systems based on iss cyclic-small-gain theorem. *Mathematics of Control, Signals, and Systems*, 24(1):75–110, 2012.

[71] S. Mahdavi, M. E. Shiri, and S. Rahnamayan. Metaheuristics in large-scale global continues optimization: A survey. *Information Sciences*, 295:407–428, 2015.

[72] M. S. Mahmoud and M. M. Hamdan. Fundamental issues in networked control systems. *IEEE/CAA Journal of Automatica Sinica*, 5(5):902–922, 2018.

[73] I. Maruta, T. Kim, D. Song, and T. Sugie. Synthesis of fixed-structure robust controllers using a constrained particle swarm optimizer with cyclic neighborhood topology. *Expert Systems with Applications*, 40(9):3595–3605, 2013.

[74] I. Maruta, T. Kim, and T. Sugie. Fixed-structure controller synthesis: A meta-heuristic approach using simple constrained particle swarm optimization. *Automatica*, 45(2):553–559, 2009.

[75] C. Meng, T. Wang, W. Chou, S. Luan, Y. Zhang, and Z. Tian. Remote surgery case: robot-assisted teleneurosurgery. In *Proceeding of the 2004 IEEE International Conference on Robotics and Automation*, volume 1, pages 819–823, 2004.

[76] Y. Minami, S. Azuma, and T. Sugie. An optimal dynamic quantizer for feedback control with discrete-valued signal constraints. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 2259–2264, 2007.

[77] Y. Minami, S. Azuma, and T. Sugie. Optimal decentralized sigma-delta modulators for quantized feedback control. *Nonlinear Theory and Its Applications, IEICE*, 3(3):386–404, 2012.

[78] Y. Minami and T. Muromaki. Differential evolution algorithm based design of discrete-valued input systems. In *Proceedings of the 2014 SICE Annual Conference*, pages 333–336, 2014.

[79] L. A. Montestruque and P. J. Antsaklis. Static and dynamic quantization in model-based networked control systems. *International Journal of Control*, 80(1):87–101, 2007.

[80] N. Moustakis, S. Yuan, and S. Baldi. An adaptive design for quantized feedback control of uncertain switched linear systems. *International Journal of Adaptive Control and Signal Processing*, 32(5):665–680, 2018.

[81] R. M. Murray, K. J. Astrom, S. P. Boyd, R. W. Brockett, and G. Stein. Future directions in control in an information-rich world. *IEEE Control Systems Magazine*, 23(2):20–33, 2003.

[82] G. N. Nair and R. J. Evans. Stabilization with data-rate-limited feedback: tightest attainable bounds. *Systems & Control Letters*, 41(1):49–56, 2000.

[83] G. N. Nair and R. J. Evans. Exponential stabilisability of finite-dimensional linear systems with limited data rates. *Automatica*, 39(4):585–593, 2003.

[84] G. N. Nair and R. J. Evans. Stabilizability of stochastic linear systems with finite feedback data rates. *SIAM Journal on Control and Optimization*, 43(2):413–436, 2004.

[85] G. N. Nair, R. J. Evans, I. M. Y. Mareels, and W. Moran. Topological feedback entropy and nonlinear stabilization. *IEEE Transactions on Automatic Control*, 49(9):1585–1597, 2004.

[86] G. N. Nair, F. Fagnani, S. Zampieri, and R. J. Evans. Feedback control under data rate constraints: An overview. *Proceedings of the IEEE*, 95(1):108–137, 2007.

[87] K. Ogata. *Modern Control Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 4th edition, 2001.

[88] P. Ögren, E. Fiorelli, and N. E. Leonard. Cooperative control of mobile sensor networks:adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302, 2004.

[89] V. K. Ojha, A. Abraham, and V. Snášel. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60(Supplement C):97–116, 2017.

[90] H. Okajima, N. Matsunaga, and K. Sawada. Optimal quantization interval design of dynamic quantizers which satisfy the communication rate constraints. In *Proceedings of the 49th IEEE Conference on Decision and Control*, pages 4733–4739, 2010.

[91] H. Okajima, K. Sawada, J. Kaibe, and N. Matsunaga. Performance improvement of dynamic quantizer with reachability of output. *Transactions of the Society of Instrument and Control Engineers*, 48(6):359–361, 2012.

[92] H. Okajima, K. Sawada, and N. Matsunaga. Dynamic quantizer design under communication rate constraints. *IEEE Transactions on Automatic Control*, 61(10):3190–3196, 2016.

[93] H. Okajima, T. Umemoto, N. Matsunaga, and S. Kawaji. Analysis of dynamic quantizer in 2-dof internal model control system with dead-time. In *Proceedings of the 2009 ICCAS-SICE*, pages 4380–4383, 2009.

[94] Y. Pan, Y. Liu, B. Xu, and H. Yu. Hybrid feedback feedforward: An efficient design of adaptive neural network control. *Neural Networks*, 76:122–134, 2016.

[95] C. Peng and F. Li. A survey on recent advances in event-triggered communication and control. *Information Sciences*, 457-458:113–125, 2018.

[96] J. Pérez, J. A. Cabrera, J. J. Castillo, and J. M. Velasco. Bio-inspired spiking neural network for nonlinear systems control. *Neural Networks*, 104:15–25, 2018.

[97] I. R. Petersen and A. V. Savkin. Multi-rate stabilization of multivariable discrete-time linear systems via a limited capacity communication channel. In *Proceedings of the 40th IEEE Conference on Decision and Control*, volume 1, pages 304–309, 2001.

[98] M. A. Potter and K. A. D. Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, pages 249–257, 1994.

[99] U. Premaratne, S. K. Halgamuge, and I. M. Y. Mareels. Event triggered adaptive differential modulation: A new method for traffic reduction in networked control systems. *IEEE Transactions on Automatic Control*, 58(7):1696–1706, 2013.

[100] K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, Berlin, Heidelberg, 1 edition, 2005.

[101] D. E. Quevedo, J. A. D. Doná, and G. C. Goodwin. Receding horizon linear quadratic control with finite input constraint set. *IFAC Proceedings Volumes*, 35(1):183–188, 2002. 15th IFAC World Congress.

[102] D. E. Quevedo, G. C. Goodwin, and J. A. De Doná. Finite constraint set receding horizon quadratic control. *International Journal of Robust and Nonlinear Control*, 14(4):355–377, 2004.

[103] D. E. Quevedo, C. Müller, and G. C. Goodwin. Conditions for optimality of naïve quantized finite horizon control. *International Journal of Control*, 80(5):706–720, 2007.

[104] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017.

[105] J. E. Rodriguez Ramirez and Y. Minami. Design of neural network quantizers for networked control systems. *Electronics*, 8(3), 2019.

[106] J. E. Rodriguez Ramirez, Y. Minami, and K. Sugimoto. Design of finite-level dynamic quantizers by using differential evolution algorithm. In *Proceedings of the 2015 SICE Annual Conference*, pages 841–844, 2015.

[107] J. E. Rodriguez Ramirez, Y. Minami, and K. Sugimoto. Design of finite-level dynamic quantizers by using covariance matrix adaptation evolution strategy. *International Journal of Innovative Computing, Information and Control*, 12(3):795–808, 2016.

[108] J. E. Rodriguez Ramirez, Y. Minami, and K. Sugimoto. Event-triggered dynamic quantizers for networked control systems. *IFAC-PapersOnLine*, 50(1):5190–5195, 2017. 20th IFAC World Congress.

[109] J. E. Rodriguez Ramirez, Y. Minami, and K. Sugimoto. Event-triggered quantizer for network traffic reduction. *Journal of Advanced Computational Intelligence & Intelligent Informatics*, 21(6):1111–1113, 2017.

[110] J. E. Rodriguez Ramirez, Y. Minami, and K. Sugimoto. Neural network quantizers for discrete-valued input control. In *Proceedings of the 11th Asian Control Conference*, pages 2019–2024, 2017.

[111] J. E. Rodriguez Ramirez, Y. Minami, and K. Sugimoto. Design of quantizers with neural networks: Classification based approach. In *Proceedings of the 2018 International Symposium on Nonlinear Theory and Its Applications*, pages 312–315, 2018.

[112] J. E. Rodriguez Ramirez, Y. Minami, and K. Sugimoto. Synthesis of event-triggered dynamic quantizers for networked control systems. *Expert Systems with Applications*, 109:188–194, 2018.

[113] M. Rădac, R. Precup, E. M. Petriu, and S. Preitl. Iterative data-driven tuning of controllers for nonlinear systems with constraints. *IEEE Transactions on Industrial Electronics*, 61(11):6360–6368, 2014.

[114] T. Sauter and M. Lobashov. End-to-end communication architecture for smart grids. *IEEE Transactions on Industrial Electronics*, 58(4):1218–1228, 2011.

[115] A. V. Savkin. Analysis and synthesis of networked control systems: topological entropy, observability, robustness and optimal control. *Automatica*, 42(1):51–62, 2006.

[116] K. Sawada, H. Okajima, N. Matsunaga, and Y. Minami. Dynamic quantizer design for mimo systems based on communication rate constraint. In *Proceedings of the 37th Annual Conference of the IEEE Industrial Electronics Society*, pages 2572–2577, 2011.

[117] K. Sawada and S. Shin. Synthesis of dynamic quantizers for quantized feedback systems within invariant set analysis framework. In *Proceedings of the 2011 American Control Conference*, pages 1662–1667, 2011.

[118] P. Seiler and R. Sengupta. Analysis of communication losses in vehicle control problems. In *Proceedings of the 2001 American Control Conference*, volume 2, pages 1491–1496, 2001.

[119] P. Seiler and R. Sengupta. An $H_\infty$ approach to networked control. *IEEE Transactions on Automatic Control*, 50(3):356–364, 2005.

[120] P. Shi, H. Wang, and C.-C. Lim. Network-based event-triggered control for singular systems with quantizations. *IEEE Transactions on Industrial Electronics*, 63(2):1230–1238, 2016.

[121] S. Shirmohammadi and N. H. Woo. Evaluating decorators for haptic collaboration over internet. In *Proceedings of the 2nd International Conference on Creating, Connecting and Collaborating through Computing*, pages 105–109, 2004.

[122] P. Y. Simard, D. Steinkraus, and J. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the 7th International Conference on Document Analysis and Recognition*, pages 958–963, 2003.

[123] R. M. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[124] C. Suh and Y.-B. Ko. Design and implementation of intelligent home control systems based on active sensor networks. *IEEE Transactions on Consumer Electronics*, 54(3):1177–1184, 2008.

[125] T. Suttorp, N. Hansen, and C. Igel. Efficient covariance matrix update for variable metric evolution strategies. *Machine Learning*, 75(2):167–197, 2009.

[126] S. Tatikonda and S. Mitter. Control over noisy channels. *IEEE Transactions on Automatic Control*, 49(7):1196–1201, 2004.

[127] S. Tatikonda and S. Mitter. Control under communication constraints. *IEEE Transactions on Automatic Control*, 49(7):1056–1068, 2004.

[128] S. Tatikonda, A. Sahai, and S. Mitter. Stochastic linear control over a communication channel. *IEEE Transactions on Automatic Control*, 49(9):1549–1561, 2004.

[129] K. Tsumura. Criteria for systems identification with quantized data and the optimal quantization schemes. *IFAC Proceedings Volumes*, 38(1):261–266, 2005. 16th IFAC World Congress.

[130] V. N. Vapnik. *Statistical Learning Theory*, chapter 9, pages 395–399. Wiley-Interscience, 1998.

[131] W.-S. Wong and R. W. Brockett. Systems with finite communication bandwidth constraints. i. state estimation problems. *IEEE Transactions on Automatic Control*, 42(9):1294–1299, 1997.

[132] W.-S. Wong and R. W. Brockett. Systems with finite communication bandwidth constraints. ii. stabilization with limited information feedback. *IEEE Transactions on Automatic Control*, 44(5):1049–1053, 1999.

[133] Y.-Q. Xia, Y.-L. Gao, L.-P. Yan, and M.-Y. Fu. Recent progress in networked control systems - a survey. *International Journal of Automation and Computing*, 12(4):343–367, 2015.

[134] L. Xing, C. Wen, H. Su, J. Cai, and L. Wang. A new adaptive control scheme for uncertain nonlinear systems with quantized input signal. *Journal of the Franklin Institute*, 352(12):5599–5610, 2015.

[135] L. Xing, C. Wen, H. Su, Z. Liu, and J. Cai. Robust control for a class of uncertain nonlinear systems with input quantization. *International Journal of Robust and Nonlinear Control*, 26(8):1585–1596, 2016.

[136] P. Yan, D. Liu, D. Wang, and H. Ma. Data-driven controller design for general mimo nonlinear systems via virtual reference feedback tuning and neural networks. *Neurocomputing*, 171:815–825, 2016.

[137] X. Yang. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[138] X. Yang. Metaheuristic Optimization. *Scholarpedia*, 6(8):11472, 2011. revision #91488.

[139] X. Yang and X. He. Firefly algorithm: Recent advances and applications. *International Journal of Swarm Intelligence*, 1(1):36–50, 2013.

[140] R. Yoshino, H. Okajima, N. Matsunaga, and Y. Minami. Dynamic quantizers design under data rate constraints by using pso method. In *Proceedings of the 2014 SICE Annual Conference*, pages 1041–1046, 2014.

[141] K.-Y. You and L.-H. Xie. Survey of recent progress in networked control systems. *Acta Automatica Sinica*, 39(2):101–117, 2013.

[142] C. Zhang and G. E. Dullerud. Finite gain stabilization with logarithmic quantization. In *2007 46th IEEE Conference on Decision and Control*, pages 3952–3957, 2007.

[143] D. Zhang, P. Shi, Q.-G. Wang, and L. Yu. Analysis and synthesis of networked control systems: A survey of recent advances and challenges. *ISA Transactions*, 66:376–392, 2017.

[144] W. Zhang, M. S. Branicky, and S. M. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21(1):84–99, Feb 2001.

[145] Y.-B. Zhao, G.-P. Liu, Y. Kang, and L. Yu. *A Brief Tutorial of Networked Control Systems*, pages 1–11. Springer Singapore, Singapore, 2018.

[146] J. Zhou. Decentralized adaptive control for interconnected nonlinear systems with input quantization. *IFAC-PapersOnLine*, 50(1):10419–10424, 2017. 20th IFAC World Congress.

# Appendix

# A. Default Parameters for the $(\mu/\mu_W, \lambda)$ CMA-ES Algorithm

The default values of these parameters were taken from [38].

i Selection and Recombination:

$$N = 4 + \lfloor 3 \ln(n) \rfloor, \quad \mu = \lfloor \mu' \rfloor, \quad \mu' = \frac{N}{2} \tag{136}$$

$$w_i = \frac{w_i'}{\sum_{j=1}^{\mu} w_j'}, \quad w_i' = \ln(\mu' + 0.5) - \ln i \quad \text{for } i = 1, \ldots, \mu \tag{137}$$

$$\mu_{\text{eff}} = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \tag{138}$$

ii Step-size control:

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 5} \tag{139}$$

$$d_\sigma = 1 + 2 \max\left(0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n+1}} - 1\right) + c_\sigma \tag{140}$$

iii Covariance matrix adaptation:

$$c_c = \frac{4 + \mu_{\text{eff}}/n}{n + 4 + 2\mu_{\text{eff}}/n} \tag{141}$$

$$c_1 = \frac{2}{(n+1.3)^2 + \mu_{\text{eff}}} \tag{142}$$

$$c_\mu = \min\left(1 - c_1, \alpha_\mu \frac{\mu_{\text{eff}} - 2 + 1/\mu_{\text{eff}}}{(n+2)^2 + \alpha_\mu \mu_{\text{eff}}/2}\right) \quad \text{with} \quad \alpha_\mu = 2 \tag{143}$$

# B. Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based metaheuristic inspired in the behavior of biological communities like swarms of bees and flocks of birds [58]. In the PSO nomenclature an individual is called *particle* and the set of all the particles is called *swarm*.

PSO is implemented very easily and has only three control parameters: inertia parameter $\chi_0$, cognition parameter $\chi_1$ and social parameter $\chi_2$. This algorithm is very good for local optimization, but not so much for global optimization. Several variations try to improve its performance and ability to *scape* from local minima. In addition, PSO is very sensitive to the tuning of the control parameters and sometimes it is difficult to find a good set of parameters to make the algorithm effective. The version used in this study is shown in Algorithm 4.

**Algorithm 4 : PSO**

**Initialization:** Given $N \in \mathbb{N}$, $k_{max} \in \mathbb{N}$, $\chi_0 \in [0,1]$, $\chi_1 \in [0,4]$, $\chi_2 \in [0,4]$ and the search spaces $S_0 = [x_{min}, x_{max}]^n$ and $V_0 = [v_{min}, v_{max}]^n$. Set $k = 0$, then select randomly $N$ individuals $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ and their velocities $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N\}$ in the corresponding search spaces.

**Step 1:** The cost function $J(\boldsymbol{\theta}_i)$ is evaluated for each $\boldsymbol{\theta}_i$. Then, the personal best solutions and the global best solution are selected by:

$$\boldsymbol{\theta}_{pbest,i} = \underset{\boldsymbol{\theta} \in \{\boldsymbol{\theta}_i^{(j)} | j=1,2,\ldots,k\}}{\arg\min} J(\boldsymbol{\theta}), \tag{144}$$

$$\boldsymbol{\theta}_{gbest} = \underset{\boldsymbol{\theta} \in \{\boldsymbol{\theta}_{pbest,i} | i=1,2,\ldots,N\}}{\arg\min} J(\boldsymbol{\theta}). \tag{145}$$

where $(j)$ indicates the generation of the respective individual. If $k = k_{max}$ then $\boldsymbol{\theta}_{gbest}$ is the solution of the algorithm, if not go to **Step 2**.

**Step 2:** Sequentially the following update laws are applied to each individual.

$$\boldsymbol{v}_i \leftarrow \chi_0 \boldsymbol{v}_i + \chi_1 \rho_{1,i} \left(\boldsymbol{\theta}_{pbest,i} - \boldsymbol{\theta}_i\right) + \chi_2 \rho_{2,i} \left(\boldsymbol{\theta}_{gbest} - \boldsymbol{\theta}_i\right), \tag{146}$$

$$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \boldsymbol{v}_i, \tag{147}$$

where $\rho_{1,i}$ and $\rho_{2,i} \in [0,1]$ are random numbers uniformly distributed. Then make $k \leftarrow k+1$ and go to **Step 1**.

# C. NNQs Statistical Analysis Details

Table 17: $E(Q_{NN})$ 3-way ANOVA for $\mathsf{h} = \text{sigm}$. For the general linear model: $E(Q_{NN})$ versus Type, Init and $n_L$. The considered significance level is $\alpha = 0.05$. The factor coding is $(-1, 0, +1)$. *There is significant difference when* P-Value$< \alpha$.

| M | Source | DF | Adj SS | Adj MS | F-Value | P-Value |
|---|---|---|---|---|---|---|
| 2 | Type | 1 | 0.0758 | 0.07576 | 0.47 | 0.496 |
| | Init | 1 | 3.2826 | 3.28257 | 20.16 | **0.000** |
| | nL | 1 | 0.2065 | 0.20646 | 1.27 | 0.261 |
| | Type*Init | 1 | 0.1295 | 0.12954 | 0.80 | 0.373 |
| | Type*$n_L$ | 1 | 0.1082 | 0.10818 | 0.66 | 0.415 |
| | Init*$n_L$ | 1 | 0.0075 | 0.00755 | 0.05 | 0.830 |
| | Type*Init*$n_L$ | 1 | 0.1229 | 0.12294 | 0.76 | 0.385 |
| | Error | 392 | 63.8188 | 0.16280 | | |
| | Total | 399 | 67.7518 | | | |

| M | Source | DF | Adj SS | Adj MS | F-Value | P-Value |
|---|---|---|---|---|---|---|
| 8 | Type | 1 | 391.470 | 391.470 | 299.31 | **0.000** |
| | Init | 1 | 24.110 | 24.110 | 18.43 | **0.000** |
| | nL | 1 | 10.096 | 10.096 | 7.72 | **0.006** |
| | Type*Init | 1 | 38.542 | 38.542 | 29.47 | **0.000** |
| | Type*$n_L$ | 1 | 2.204 | 2.204 | 1.68 | 0.195 |
| | Init*$n_L$ | 1 | 10.787 | 10.787 | 8.25 | **0.004** |
| | Type*Init*$n_L$ | 1 | 6.449 | 6.449 | 4.93 | **0.027** |
| | Error | 392 | 512.695 | 1.308 | | |
| | Total | 399 | 996.352 | | | |

Table 18: Tukey pairwise comparison 3-way ANOVA for $\mathsf{h}$ = sigm and $M = 2$. Grouping Information Using the Tukey Method and 95% confidence. *Means that do not share a letter are significantly different.*

| Factor | | N | Mean | Grouping | | |
|---|---|---|---|---|---|---|
| Type*Init | C Urand | 100 | 4.28598 | A | | |
| | R Urand | 100 | 4.27751 | A | B | |
| | R Xavier | 100 | 4.13233 | | B | C |
| | C Xavier | 100 | 4.06881 | | | C |
| Type*$n_L$ | R L2 | 100 | 4.24408 | A | | |
| | C L2 | 100 | 4.18367 | A | | |
| | C L4 | 100 | 4.17112 | A | | |
| | R L4 | 100 | 4.16575 | A | | |
| Init*$n_L$ | Urand L2 | 100 | 4.30012 | A | | |
| | Urand L4 | 100 | 4.26337 | A | B | |
| | Xavier L2 | 100 | 4.12763 | | B | C |
| | Xavier L4 | 100 | 4.07350 | | | C |
| Type*Init*$n_L$ | R Urand L2 | 50 | 4.32987 | A | | |
| | C Urand L4 | 50 | 4.30158 | A | | |
| | C Urand L2 | 50 | 4.27038 | A | B | |
| | R Urand L4 | 50 | 4.22516 | A | B | |
| | R Xavier L2 | 50 | 4.15830 | A | B | |
| | R Xavier L4 | 50 | 4.10635 | A | B | |
| | C Xavier L2 | 50 | 4.09696 | A | B | |
| | C Xavier L4 | 50 | 4.04066 | | B | |

Table 19: Tukey pairwise comparison 3-way ANOVA for $\mathsf{h}$ = sigm and $M = 8$. Grouping Information Using the Tukey Method and 95% confidence. *Means that do not share a letter are significantly different.*

| Factor | | N | Mean | Grouping | | |
|---|---|---|---|---|---|---|
| Type*Init | C Urand | 100 | 2.86371 | A | | |
| | C Xavier | 100 | 2.73390 | A | | |
| | R Xavier | 100 | 1.37617 | | B | |
| | R Urand | 100 | 0.26432 | | | C |
| Type*$n_L$ | C L2 | 100 | 2.88345 | A | | |
| | C L4 | 100 | 2.71416 | A | | |
| | R L2 | 100 | 1.05333 | | B | |
| | R L4 | 100 | 0.58715 | | | C |
| Init*$n_L$ | Xavier L2 | 100 | 2.37812 | A | | |
| | Xavier L4 | 100 | 1.73195 | | B | |
| | Urand L4 | 100 | 1.56937 | | B | |
| | Urand L2 | 100 | 1.55866 | | B | |
| Type*Init*$n_L$ | C Urand L2 | 50 | 2.91111 | A | | |
| | C Xavier L2 | 50 | 2.85580 | A | | |
| | C Urand L4 | 50 | 2.81630 | A | | |
| | C Xavier L4 | 50 | 2.61201 | A | | |
| | R Xavier L2 | 50 | 1.90045 | | B | |
| | R Xavier L4 | 50 | 0.85188 | | | C |
| | R Urand L4 | 50 | 0.32243 | | | C |
| | R Urand L2 | 50 | 0.20622 | | | C |

Table 20: $E(Q_{NN})$ 3-way ANOVA for different activation functions ($M = 8$). For the general linear model: $E(Q_{NN})$ versus Type, Init and h. The considered significance level is $\alpha = 0.05$. The factor coding is $(-1, 0, +1)$. *There is significant difference when* P-Value$< \alpha$.

| Source | DF | Adj SS | Adj MS | F-Value | P-Value |
|---|---|---|---|---|---|
| Type | 1 | 248.95 | 248.947 | 173.20 | **0.000** |
| Init | 1 | 13.61 | 13.610 | 9.47 | **0.002** |
| h | 2 | 177.47 | 88.735 | 61.74 | **0.000** |
| Type*Init | 1 | 11.71 | 11.714 | 8.15 | **0.004** |
| Type*h | 2 | 295.36 | 147.678 | 102.75 | **0.000** |
| Init*h | 2 | 77.41 | 38.704 | 26.93 | **0.000** |
| Type*Init*h | 2 | 69.32 | 34.659 | 24.11 | **0.000** |
| Error | 1188 | 1707.53 | 1.437 | | |
| Total | 1199 | 2601.36 | | | |

Table 21: 3-way ANOVA Tukey pairwise comparison for different activation functions ($M = 8$). Grouping Information Using the Tukey Method and 95% confidence. *Means that do not share a letter are significantly different.*

| Factor | | N | Mean | Grouping | | | |
|---|---|---|---|---|---|---|---|
| Init*h | Xavier ReLU | 200 | 2.56526 | A | | | |
| | Xavier sigm | 200 | 2.05504 | | B | | |
| | Urand ReLU | 200 | 1.91764 | | B | | |
| | Urand sigm | 200 | 1.56401 | | | C | |
| | Urand tanh | 200 | 1.55034 | | | C | |
| | Xavier tanh | 200 | 1.05068 | | | | D |
| Type*Init*h | R Xavier ReLU | 100 | 2.99436 | A | | | |
| | C Urand sigm | 100 | 2.86371 | A | | | |
| | C Xavier sigm | 100 | 2.73390 | A | | | |
| | C Xavier ReLU | 100 | 2.13617 | | B | | |
| | C Urand ReLU | 100 | 1.93525 | | B | | |
| | R Urand ReLU | 100 | 1.90003 | | B | C | |
| | C Urand tanh | 100 | 1.89586 | | B | C | |
| | C Xavier tanh | 100 | 1.87093 | | B | C | |
| | R Xavier sigm | 100 | 1.37617 | | | C | D |
| | R Urand tanh | 100 | 1.20482 | | | | D |
| | R Urand sigm | 100 | 0.26432 | | | | E |
| | R Xavier tanh | 100 | 0.23043 | | | | E |