Pontifícia Universidade Católica
do Rio de Janeiro

Rafael Albert Silva de Medeiros

Application of the Keras Framework to Nonlinear Black-Box System Identification of Piezoelectric Micromanipulators

Graduation Project

Graduation Project presented to the department of Mechanical Engineering of PUC-Rio.

Advisor: Helon Vicente Hultmann Ayala

Rio de Janeiro
December 2018

Acknowledgment

Agradeço a PUC-Rio, que me deu oportunidade para expandir meus horizontes e ao meu orientador Helon Ayala pela oportunidade e suporte neste projeto.

Agradeço aos meus pais Rosélia e Alberto e meus avós Marlene e Adalberto por todo o suporte que me deram durante esta jornada.

Agradeço a minha namorada Thallita, pelo suporte e pela compreensão nos momentos em que estive distante.

E para todos aqueles que estiveram envolvidos direta ou indiretamente em minha graduação, meu muito obrigado.

Abstract

Application of the Keras Framework to Nonlinear Black-Box System Identification of Piezoelectric Micromanipulators

Micromanipulation has several applications in different fields of science. This document aims to use the Keras framework as a system identification tool combining with artificial neural networks to solve problems related to micromanipulation with 1 DoF and 2 DoF.

Keywords

Neural Network; Micromanipulation; System Identification; Keras;

Summary

List of Figures

List of Tables

1 INTRODUCTION

System identification is the use of inputs and output data of a certain system to create a data-driven model to simulate the system. This model can be used for example as a feedback for micromanipulators. System identification is the use of inputs and outputs of measured data of a certain system to create a mathematical model that represents the system observed (Chen, Billings & Grant, 1990).

Micromanipulators are devices widely used in medical tasks for the manipulation of microscopic level samples under a microscope and for tissue examination and dissection. The micromanipulators, when interacting with the samples, need to have a high precision. For an interaction with the sample the micromanipulator has a gripper that can be fabricated with piezoelectric material. When applying a small electrical discharge the gripper closes, capturing the sample.

The technological development of artificial intelligence is a relative young branch of science (Cohen & Feigenbaum, 2014). However it already have applications in several current technologies, the mankind realizes this powerful tool can be useful in solving repetitive problems that an ordinary machine was not capable of doing, or problems that even the human being was not physically capable of solving. So, in order to create a machine that could learn how humans learn, artificial neural networks have been created to have their development based on how neurons transmit information. Artificial neural networks, unlike a standard computational algorithm, are able to learn new functions and have multiple processing in parallel. Because parallel processing was slow, neural networks have long been an unattractive field of study, but this has changed with the advent of GPUs (graphics processing units) that can quickly solve the problems in parallel like the network need (Oh & Jung, 2004). After the introduction of the GPUs in the market in an accessible way, artificial intelligence took a great leap and began to be implemented in the most diverse technologies, ranging from autonomous cars until its use in the medicine of dissection tissues using micromanipulators. Like the brain neurons, neural networks have processing cores that are divided into layers. For networks with many layers the approach is made by deep learning and are called deep neural networks (Goodfellow et al., 2016)

An interesting field of application of deep learning is the identification of dynamic systems. Using the inputs and outputs of the system, the tool creates a model that represents, ideally with high precision, the dynamics of the system studied.A possibility to create a model is using the Keras framework, an open source neural network library, designed to enable fast experimentation with deep neural networks. Keras framework has high performance and an ease training on large datasets (Chollet & Allaire, 2018). The use of deep learning applied to system identification has several advantages such as its simplicity, high accuracy with respect to the real system and its high capability of recognition and classification of patterns (Schmidhuber, 2015). It is also a interesting tool for modeling non-linear systems. With the model of the system reproduced by the system identification tool, it is possible to obtain feedback for application in micromanipulation systems, which requires a high precision, whenever the output is assumed not to be measured. Moreover piezoelectric materials, used in grippers of micromanipulation, have some intrinsic problems like hysteresis, creep and coupling effect, making manipulation difficult without the use of a feedback system that regulates these problems. So the use of deep learning applied to systems identification is able to improve the accuracy of micromanipulators through their feedback and so with greater precision the equipment may be able to perform more precise positioning tasks.

1.1 Motivation

Recently the use of deep learning is increasing, associated with new technologies coming, its use will increase widely in the next few years. The increased use of this technology is due to its large application, ease of use and primarily because of the amount of data increasing largely day after day. Nowadays all big companies as Google, Netflix, NASA and others, already uses deep learning in some way in their technologies, but in general the large amount of data need to be treated with some smart technology as deep leaning (Genc, 2017). The big companies which don't plan to use this kind of treatment in their data, will have problems in competing with companies that use such technology. So the project done in this document will enhance the field of application, improving feedback for actual and future systems in order to solve problems associated with precision control.

1.2 Objectives

The main goal of this project is to obtain models with nonlinear black-box system identification using artificial neural networks. This can be used to provide feedback on several kinds of different systems. One of the main system of interest is the piezoelectric micromanipulation, so the feedback will make micromanipulation more precise, and the user can be able to perform the increasingly complex tasks and area of operation of the micromanipulation devices.

1.3 Literature Review

The project here presented aims to expand the way in which micromanipulation can obtain a precise feedback through neural networks for solving the problems of micromanipulation e.g. hysteresis, creep and coupling effect, by application of recently developed tools for creating artificial neural network models. According to (Rakotondrabe, 2013), the hysteresis associated to smart materials need the use of a feedback control to compensate the nonlinearities in micromanipulation. In (Goldfarb & Celanovic, 1997), the authors models a piezoelectric stack actuators to control micromanipulation systems which require an accurate position and force control. For (Zhang, Han, Yu, Shee & Ang, 2012), the modeling of micromanipulation problems can be done through an automatic Prandtl-Ishlinskii method, using for this the vision-feedback. The advantage of micromanipulation for biomedical applications instead the common manually based injection methods is shown in (Tan & Ng, 2001), while in (Nakayama et al., 1999) is shown an example of application of micromanipulation: improve human fertilization in vitro.

The use of artificial neural networks as a black-box system identification to build a model for micromanipulation is challenging, due to the hysteresis problem of piezoelectric materials. For this purpose, in (Noël, Esfahani, Kerschen & Schoukens, 2017) the authors uses a black-box based in Bouc-Wen equations to model the hysteresis while in (Habineza, Rakotondrabe & Le Gorrec, 2015), (Aljanaideh & Rakotondrabe, 2018) and (Rakotondrabe, 2017) the authors use models of 2-DoF with different approaches for modelling and control of a multivariable hysteresis in piezoelectric systems to get a feedback. A nonlinear black-box system identification for modelling the 2-DoF micromanipulation problems is discussed in (Ayala, Rakotondrahe & Coelho, 2018).

The authors in (Chen et al., 1990) and (Žilková, Timko & Girovskỳ, 2006), shows neural networks as black-box and they also show the advantages of the use of neural network in system identification. In (Haykin, 2009) the author show how neural networks can be used in regression to solve problems of large datasets. In (De la Rosa & Yu, 2016) the authors use nonlinear system identification with deep learning modification to construct the statistical features of the hidden weights, improving the search for weights in regression and classification problems. And finally in (Genc, 2017) the author talks about the scalability and robustness issues in complex nonlinear system identification of dynamic system within the framework of deep convolutional neural networks (CNNs) applying in Keras with Tensorflow backend, and to building energy load prediction problem.

2 METHODS

The chapter is organized as follows: section 2.1 explain how to model the nonlinear system of micromanipulation with artificial neural networks. Section 2.2 explain what is the Keras framework and how it works. Section 2.3 shows the validation of the proposed model. Section 2.4 explain the pre-processing done on the data. Section 2.5 shows a case of study using a piezoelectric micromanipulator with 1 DoF.

2.1 Nonlinear System Identification with Artificial Neural Networks

Artificial Neural Networks (ANN's) have an interesting ability to model nonlinear systems (Žilková et al., 2006). Therefore, ANN are a good tool to model nonlinear problems of micromanipulation (e.g. the hysteresis).

Using ANN to model (De la Rosa & Yu, 2016) a nonlinear system (Figure 2.1), the follow equation is obtained:

$$\begin{aligned}\hat{y}(t) = \beta \cdot \phi_p \cdot (W_p \cdot \phi_{p-1} \cdot (... + W_3 \cdot \phi_2 \cdot (W_2 \cdot \phi_1(W_1 \cdot (u(t-1) + ... + u(t-n) + \\ + y(t-1) + ... + y(t-n)) + b_1) + b_2) + b_3 + ...) + b_p)\end{aligned} \tag{2-1}$$

Where $\hat{y}$(t) is the output of the neural model, p is the number of hidden layers, l_i with (i = 1...p) are the number of neurons in each layer, n is the order of the regression, W_i are the weight, b_i are the bias, ϕ_i are the active function and β is the weight of the output layer.

A neural model need to find the best representation of the system, for this it is necessary to reduce the residual between the output of the model and the output of the system:

$$e(k) = \hat{y}(k) - y(k) \tag{2-2}$$

Where y(k) is the output of the system.

In order to build an ANN, it is necessary to choose a network architecture. When constructing ANNs, one of the main considerations is the choice of activation functions. This is because calculating the backpropagated error signal that is used to determine ANN parameter updates requires the activation function gradient. Some of the most commonly used activation functions in

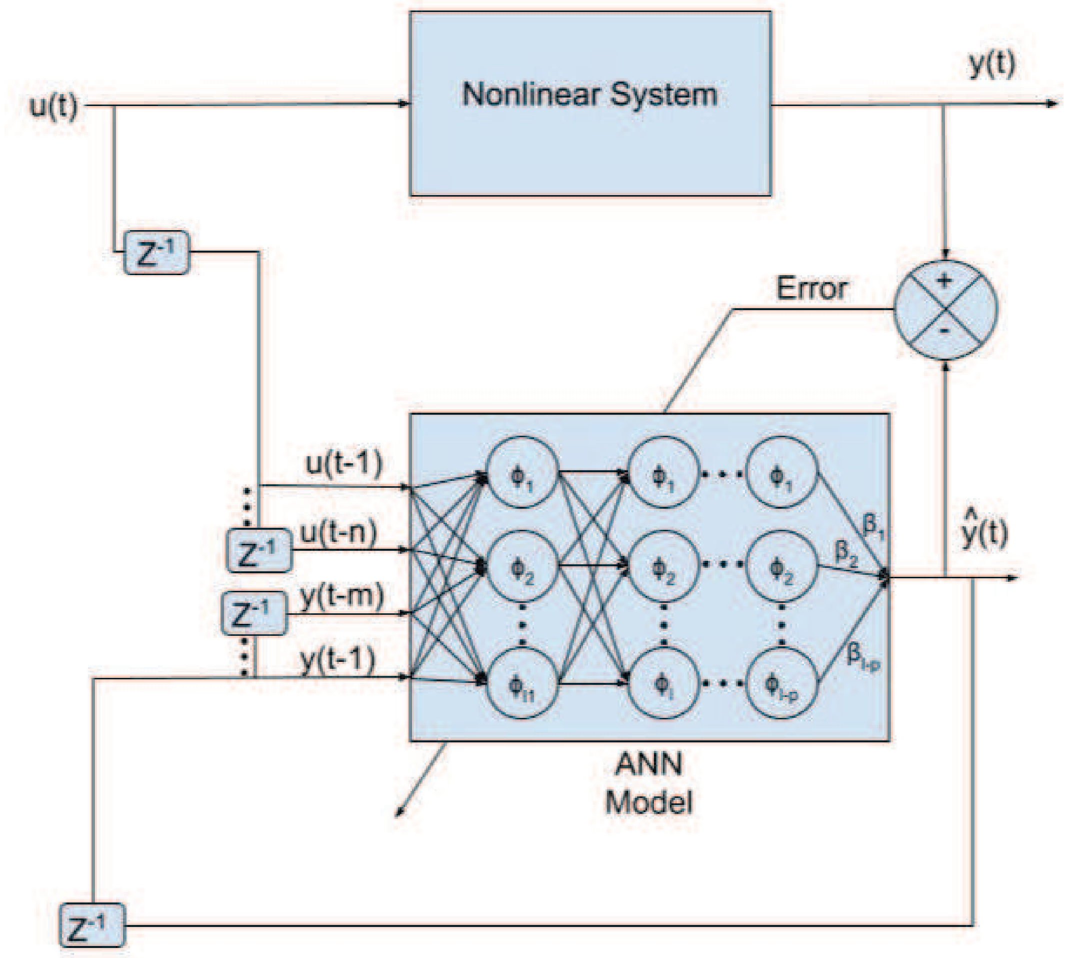

Figure 2.1: Illustration of ANN Model used for solve 1-DoF problems, where z^{-1} is a delay in the order of the regressor.

ANNs are the Uni-polar sigmoid, Bi-polar sigmoid, Conic Section, Radial Bases Function (RBF), and the hyperbolic tangent (tanh) function (Karlik & Olgac, 2011). The activation functions are used to determine if the output of the neural network has a value sufficient to activate a neuron output or not. To obtain the described models, the hyperbolic tangent function (tanh) was used. The parameters to be adjusted within the network are called weights, they are associated with each neuron and are used to give more or less importance on information that travels through the network. The weights are used to create a model that represents the real system.

Another important parameter to build an ANN is the type of training. There are 3 different models of ANN training: supervised learning, unsupervised learning and reinforcement learning (Haykin, 2009). In this project will be employed supervised learning, where the network receives several input samples and its expected final result, so the output given by the model is compared to the expected output and its residual will be used to adjust the network parameters.

A problem associated with ANN is the definition of the values of weights. Sometimes the back-propagation can not find a suited weight value due to the well-known differential problems in finding the minimum local into a function. In these cases, it becomes impossible to solve the problem manually.

In general ANNs has many layers and each layer contains many neurons, therefore it becomes impossible to manually adjust each of these weights. The learning algorithm that adjust the weights is called back-propagation. In a summarized way, back-propagation is a gradient descendent that finds the best value for each weight in order to obtain a satisfactory model with respect to the real system. For this, the back-propagation need to use the residuals. The metric based on residual is called cost function.

Gradient descent (GD) is an optimizer, i.e. it will determine how ANN will be updated based on the cost function. Gradient descent works in a multidimensional space of weights, bias and cost function. In this way it searches the local minimum through the inverse gradient, i.e. since the gradient of a function shows the direction of greatest growth at one point of a curve, the inverse gradient will look for the direction of greatest decay at this point. When adding training examples to train the ANN, the gradient descent looks for the local minimum in relation to the weights and biases and the backpropagation is in charge of updating these parameters in the network. The Figure 2.2 show in an abstract way how the gradient descent works.

To train the network for each example at a time is very slow, then an alternative way through the called mini-batches was developed. This way the gradient does not lead directly to the local minimum, but through curves on the surface of the function, the gradient converges faster to the local minimum. When using the mini-batch the gradient descent gains the name of Stochastic Gradient Descent (SGD) due to its way of reaching the local minimum. The Figure 2.3 show in an abstract way how the gradient descent works.

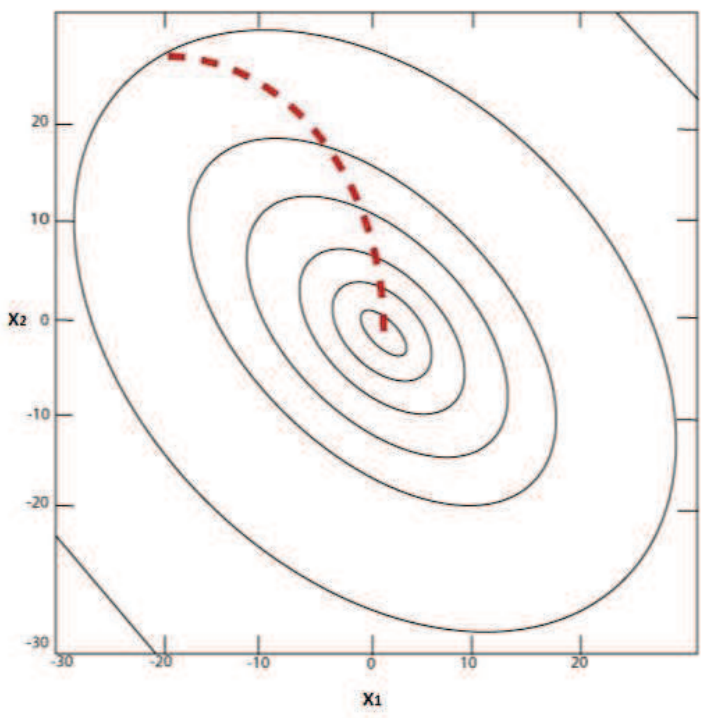

Figure 2.2: Gradient Descent converging to local minimum, x_1 and x_2 represents weights or bias (Goodfellow et al., 2016).

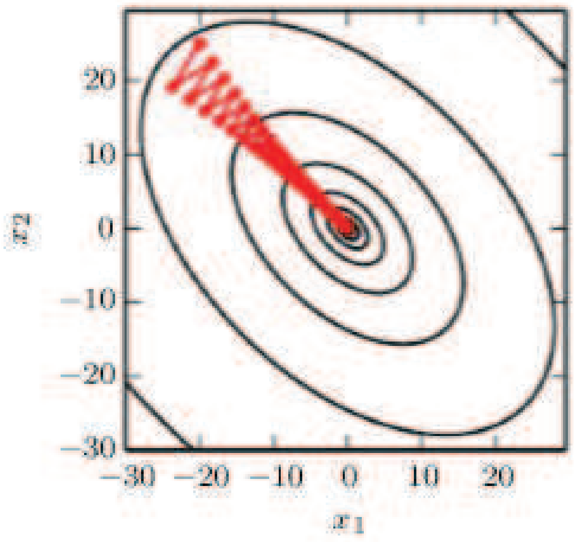

Figure 2.3: Stochastic Gradient Descent converging to local minimum, x_1 and x_2 represents weights or bias (Goodfellow et al., 2016).

In this project the optimization algorithm of gradient descent used is the RMSProp. RMSprop, proposed by Geoff Hinton, is an unpublished adaptive learning rate method that optimizes the gradient descent making the convergence become faster. RMSprop adapts the leaning rate to slow down as it approaches the minimum location and grows away from the minimum location. In this way it becomes suitable to deal with sparse data, improving SGD performance. a good default value for the learning rate, using RMSprop, is 0.001. The RMSProp has been shown effective in solve problems of neural networks, being widely used by the scientific community (Goodfellow et al., 2016).

When using ANN, usually the number of data is large. It is computationally expensive to use all the data to train your algorithms at the same time. Instead, it is more practical to compute it by splitting them into randomly small samples from the dataset, this is called mini-batch (Goodfellow et al., 2016). Thus, the batch size is the number of data in each portion, in which all are processed simultaneously in a large batch. In addition to increasing training speed, the mini-batch also avoids repeated training samples by reducing computational cost. when using small batch size the time of processing will increase, because will take more time to process all the data.

Each time a mini-batch ends, an iteration ends. After going through all the mini-batches, an epoch is completed. Normally, the training should last for many epochs, so the network fits well to the data. However if there are too many epochs an overfitting can occur. Another important parameter is the learning rate that controls the speed of adjustment of the network weights in relation to the loss gradient. The lower the value, the slower the learning. So by using a low learning rate to ensure that the best value is not lost, it will lead to significant time to converge, and sometimes problems occur when the learning rate is too low and can be stuck in a bad value (Chollet & Allaire, 2018).

2.2
Keras framework

The system identification will be done through the Keras framework within the R language. Released in 2015, Keras is an open source neural network library written in Python language designed to have ease of use, accessibility and enable good performance of neural networks, giving possibility to train almost any kind of deep-learning model.

Keras is very popular between academic research and engineers. With more then 150,000 users Keras dominates the competitions of Kaggle, a machine learning competition website, where the majority of participants were made your deep learning using Keras Framework 13. Figure 2.4 show Keras as position number two in mentions in scientific papers, only behind of Tensorflow.

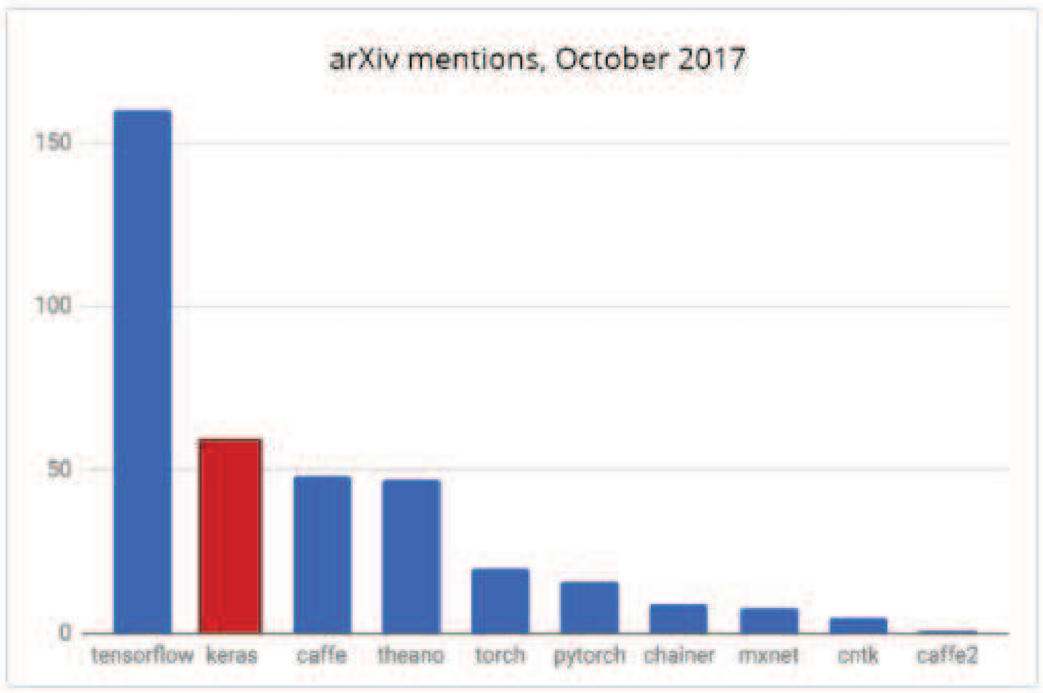

Figure 2.4: Position of Keras in mentions of scientific papers.

It's important to note the two most popular languages in data science, R and Python, are supported by Keras. The performance and the ease of training large datasets made even the European Organization for Nuclear Research (CERN), NASA, Google, Netflix and Uber adopt it to solve wide range of problems. Finally a recent big step, given by LEGO, is the use of Keras to make deep learning easy in manipulation of LEGO bricks which made a revolution in the way of teach students to use neural networks in high school (Chollet & Allaire, 2018).

2.3 Model validation

To test the model, the one-step-ahead (OSA) simulation was used. With OSA, the program simulates always one step forward of the data, i.e, based on the number of regressors, the program takes a number of samples equal the number of regressors, then simulates the output for one step ahead of the last regressors taken from output sample. Next the program takes the same number of data, but with one new sample forward in the data (instant of regression t) to simulates the next output (Chen et al., 1990).

$$\hat{y}(t) = F[y(t-1), y(t-2), ..., y(t-ny), u(t-1), u(t-2), ..., u(t-nu)] \quad (2\text{-}3)$$

Where $F[\cdot]$ is a nonlinear function given by the ANN model using Keras Framework. y is a measured output while $\hat{y}$ is a predicted output. And t is the instant of the regression.

To evaluate the quality of the model was used the free-run simulation in which the sample is used only to define the initial conditions of the model and the model itself at each step is based on the previously calculated step. The number of samples is equal the number of regressors chosen (Ayala et al., 2018):

$$\begin{cases} \hat{y}(1) = y(1) \\ \hat{y}(2) = y(2) \\ \quad\quad . \\ \quad\quad . \\ \quad\quad . \\ \hat{y}(ny) = y(ny) \\ \hat{y}(t) = F[\hat{y}(t-1)\,,\,\hat{y}(t-2)\,,\,...\,,\,\hat{y}(t-ny)\,,\,u(t-1)\,,\,u(t-2)\,,\,...\,,\,u(t-nu)] \end{cases}$$

In order to determine whether the values obtained are reasonable, the multiple correlation coefficient can also be used. The multiple correlation coefficient, also called R^2, is a measure of adjustment model, in relation to the observed values. The R^2 ranges from $-\infty$ to 1. The higher the R^2, the more explanatory the model is, and better it fits the given sample. The multiple correlation coefficient is given by:

$$R^2 = \frac{1 - \sum_{i=1}^{N} \epsilon^2}{\sum_{i=1}^{N} (y - \overline{y})^2} \quad (2\text{-}4)$$

Where: ϵ is the residual and $\overline{y}$ is the average of the outputs of the sample.

2.4 Case study: 1-DOF

The first case of study in this project will be the piezoelectric micromanipulator with 1 DoF. The principle of the micromanipulator is a cantilever made of two layers of different juxtaposeds material. One layer is called active layer and is made of a piezoelectric material. The other layer is called passive layer and is made of a non-piezoelectric material (Figure 2.5(a)). When a voltage is applied on the micromanipulator, the active layer expands a while the passive remains the same size, then the active layer bends to the side of passive layer (Figure 2.5(b)) (Ayala et al., 2015)

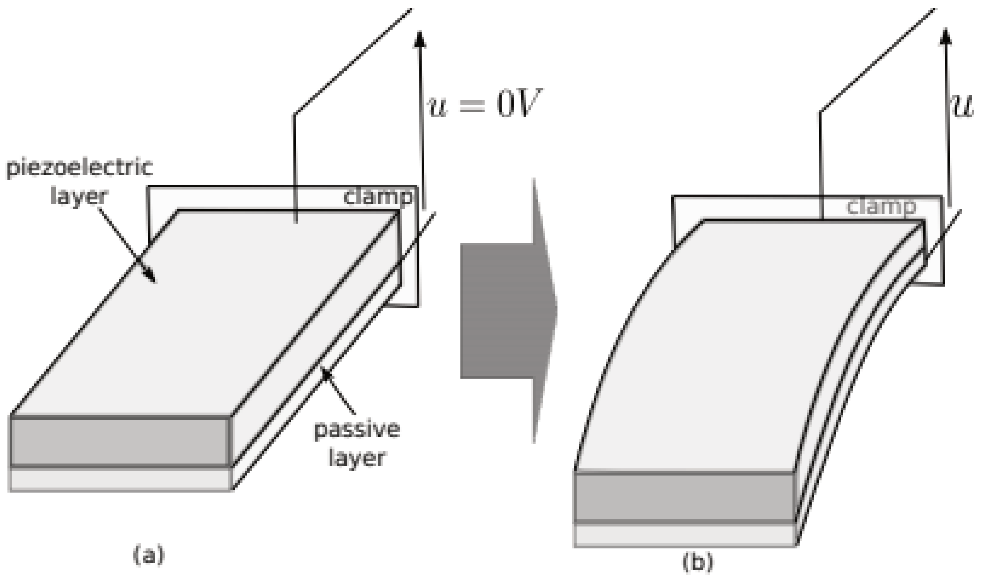

Figure 2.5: Piezoelectric cantilever bending with the imposition of a voltage. (Ayala et al., 2015)

The micromanipulator, as previously explained, has certain drawbacks such as hysteresis, creep and coupling effect that decrease the accuracy of the equipment and make a difficult system to model. The hysteresis problem appears in the model of 1 degree of freedom . This hysteresis creates a path-dependent memory effect where output depends not only on the current state but also on the previous state. The hysteresis problem is that beside of being a non-linear system, it limits the performance of piezoelectric actuators considerably because it causes fluctuations in system responses and creates poor tracking performance and potential instabilities. The creep problem also appears in a 1 DoF model and can occur during slow or fast operations. These creep effects produce significant loss in accuracy when positioning is absolutely needed for long periods of time and high oscillations at high excitation frequencies (Rakotondrabe, 2013). Thus hysteresis and creep effects must be modelled, so a feedback system is able to increase accuracy and make use of the micromanipulator more accurate.

Because of the nonlinearities, the micromanipulator can be well modeled by a nonlinear neural network black-box model for system identification showed in this project. In this way, this project has 50000 samples equally divided for training (25000 samples) and for testing (25000 samples). In figure 2.6 is shown the hysteresis curve measured at the input voltage and output displacement.

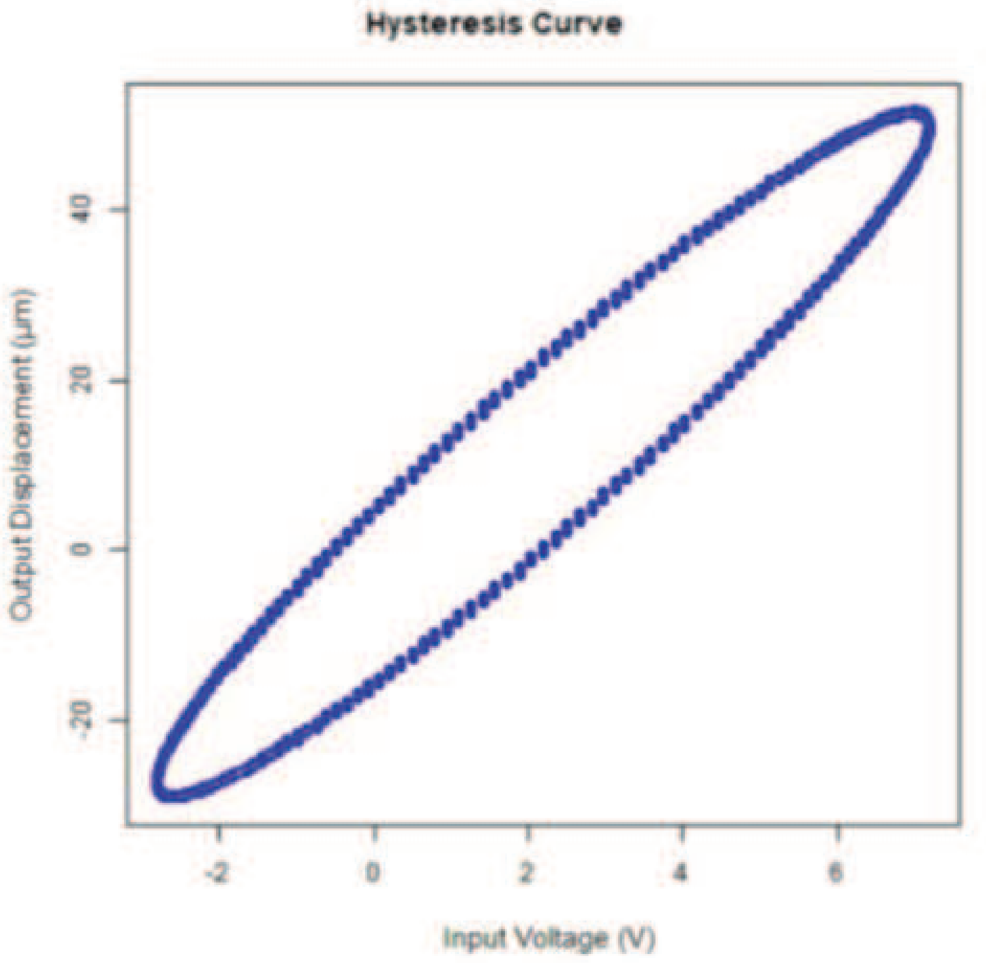

Figure 2.6: Hysteresis plot for the sinusoidal input.

2.5 Case study: 2-DOF

The second case of study in this project will be the piezoelectric micromanipulator with 2 DoF. In a 2 DoF model, besides of hysteresis and creep effect, it have also the coupling effect. This drawback presented in piezoelectric actuator model of 2 DoF, occurs by a movement of the actuator in two different degrees of freedom at the same time, i.e. when applying a voltage in order to move the actuator in the direction x, it will move as required in the x direction, but the coupling effect will make it move in the y direction as well (Figure 2.7). Thus, in a 2 DoF micromanipulator, hysteresis, creep and coupling effect must be modelled, so a feedback system is able to increase the accuracy of the micromanipulator.

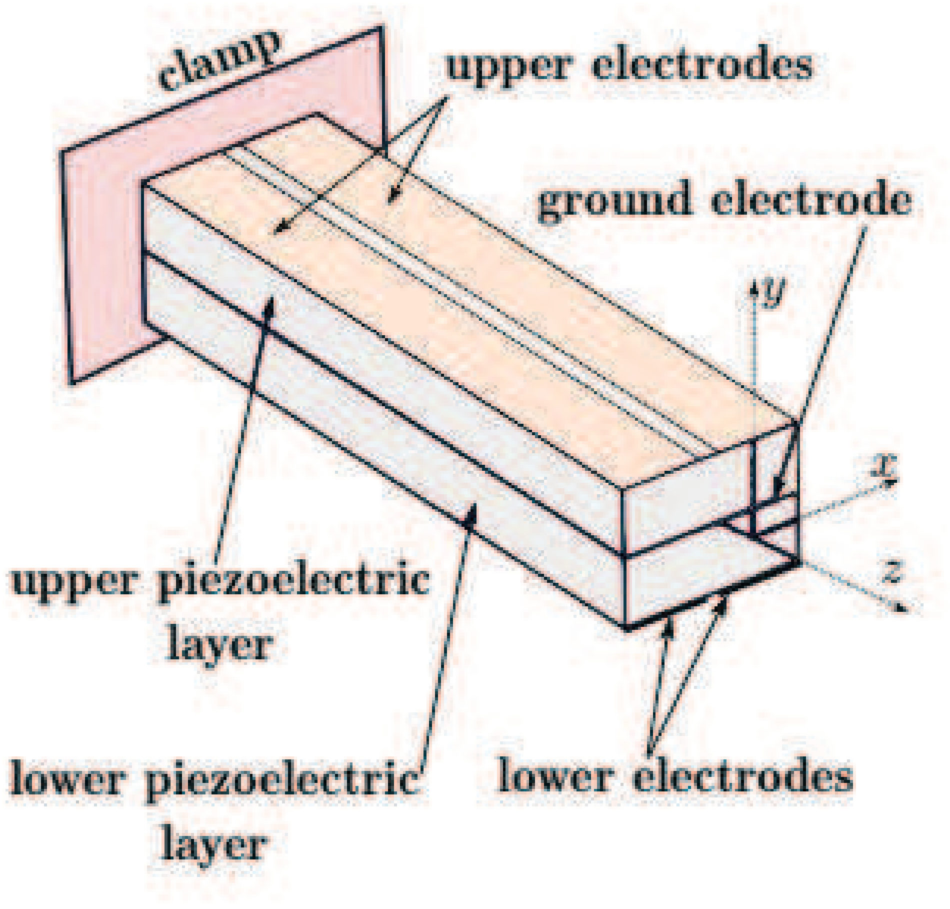

Figure 2.7: Piezoelectric cantilever axis. (Rakotondrabe, 2017)

The 2 DoF case can also be modeled by a nonlinear neural network black-box model for system identification. The case dispose of 500 samples equally divided for training (250 samples) and test (250 samples). Also the it has 2 inputs (voltage) and 2 outputs (displacement in x and y direction). The model was divided into two parts. For each output, both the input were used into the model. Then 2 models were made for the outputs. This could be done because the outputs are independent.

3 RESULTS

The objective of the project is to apply nonlinear system identification through the Keras framework to find a suitable model for a micromanipulator with 1 and 2 DoF . In order to obtain a feedback that considerably corrects its hysterical behaviour, creep and coupling effect, this model will be used to provide better information for the regulator. Therefore it will increase its accuracy and will simplify the use of micromanipulators making possible to solve more complex tasks.

The project uses the system identification and ANN to find an acceptable model for micromanipulator problems. The system identification uses previously values obtained and re-inserted them into the network to help improve the fit of the ANN model. When looking for the number of regressors that fits well to the data, it is possible to find different numbers. However, as the number of regressors increases, the computational complexity increases, leading to a higher operating cost, i.e. a longer time until the program finishes executing. So it is necessary to establish a minimum number of regressors in order to obtain reasonably and quickly results. However there is no method for directly finding the best model for the problem (Knerr, Personnaz & Dreyfus, 1990).

The structure of this chapter is organized as follows: section 3.1 explain how the computational experiences were obtained for 1 DoF. Section 3.2 explain the metrics used to chose the best suitable model for 1 DoF. Section 3.3 explain how the computational experiences were obtained for both outputs for 2 DoF and the the metrics used to chose the best suitable model for each output.

3.1 Computational Experiments for 1 DoF

Nonlinear AutoRegressive with eXogenous inputs model (NARX) with ANNs were tested by varying the order of the regressors in both input and output (both with the same order always) in order to check how the accuracy of the model changes by gradually increasing the number of regressors and neurons. The first step was define the number of regressors ranging from 2 to 5 and making a combination between the regressors and the number of neurons, which are ranging from 6 to 12 with step 2, thus creating 16 different models. It is important to note that all models made here were made with only 1 deep layer and with 100 epochs. Each combination of regressors and neurons was made for two learning rates: 0.01 and 0.001 and combining these learning rates with two different batches: 1 and 128. In this way, was possible to obtain 64 models (figure 3.1).

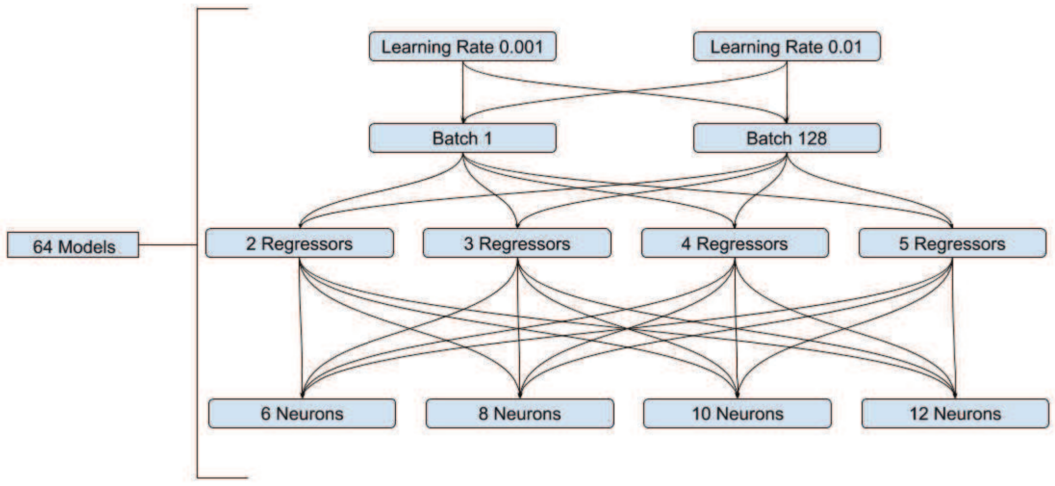

Figure 3.1: Combination of parameters used in this document to form 64 models used in 1-DoF problem

For all the models, we saved the following correlation coefficients R^2: for One-step-ahead (OSA) and Free-run (FR), both for validation and training phases, in order to compare all of them and choose the best result of model. Another important parameter to choose the best model is the time of operation to build the model through training. As explained previously, the operating cost need to be taken into account. Then, beyond the R^2, was saved also the elapsed time to build the model through training and we took the average time of all the models for each combination of batch and learning rate, this combination we called the cases of the 1 DoF modelling. The average time of all models in each table are written at the top of their respective table (case). Also, after the table, is printed the 3D graph for a better visualization of the models. The axis of the graph are regressors, neurons and elapsed time to build that specific model only.

1 Layer; Batch 1; Learning Rate 0.001; 50.000 Samples (Case 1)				
Average Time 1h 43 min				
R^2, OSA Training				
Regressors/Neurons	6	8	10	12
2	0.9997077	0.9996126	0.9994604	0.9997091
3	0.9983698	0.9994410	0.9994927	0.9995370
4	0.9982744	0.9990404	0.9997122	0.9996179
5	0.9989511	0.9989521	0.9995515	0.9995729
R^2, FR Training				
Regressors/Neurons	6	8	10	12
2	0.8360115	0.7773966	0.9992819	0.9962369
3	0.8701254	0.6272992	0.9447296	0.9934010
4	0.8742571	0.9816253	0.6340712	0.4482929
5	0.4494410	0.2534791	0.7704910	0.8378925
R^2, OSA Test				
Regressors/Neurons	6	8	10	12
2	0.9996992	0.9995737	0.9994455	0.9996864
3	0.9983558	0.9994381	0.9995026	0.9995049
4	0.9981580	0.9989883	0.9997258	0.9996378
5	0.9989541	0.9989392	0.9995577	0.9995628
R^2, FR Test				
Regressors/Neurons	6	8	10	12
2	0.8350828	0.7767934	0.9992923	0.9963151
3	0.8691701	0.6268060	0.9450209	0.9935666
4	0.8738049	0.9815459	0.6357256	0.4484921
5	0.5725771	0.2522576	0.7707644	0.8473604

Table 3.1: Multiple Correlation Coefficients for Batch 1 and Learning Rate 0.001 (1 DoF).

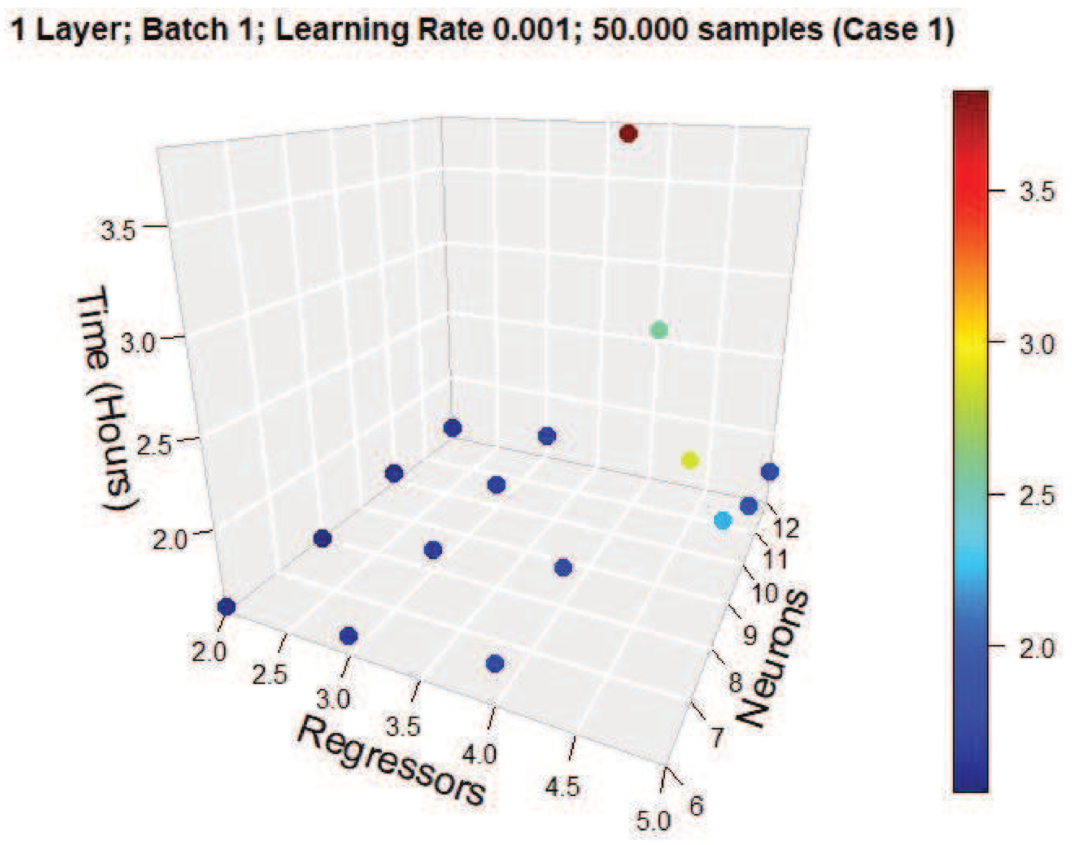

Figure 3.2: 3D Graph showing how long take to build the model with the combination of regressors, neurons and parameters .

For the case 1, it is possible to see that all R^2 values of OSA for training and test are greater than 0.9 in any combination of regressors and neurons. Otherwise the R^2 for FR in training and test got some values less than 0.9 and even less than 0.3, however all the values are positive. The majority of values of R^2 in FR greater than 0.9 became concentrated in a region which has the greater number of neurons (10 and 12) and the lowest number of regressor (2 and 3). Note that the average time to compile this package of models was 1 hour and 43 minutes.

In the graph is showed that with less neurons and regressors the elapsed time to build the model is lower with exception of 2 points out of the curve.

1 Layer; Batch 128; Learning Rate 0.001; 50.000 Samples (Case 2)				
Average Time 24 min				
R^2, OSA Training				
Regressors/Neurons	6	8	10	12
2	0.9996278	0.9997041	0.999746	0.9997546
3	0.9993154	0.9995368	0.9996898	0.9997338
4	0.9993794	0.9994317	0.9997598	0.9996422
5	0.9989707	0.9995576	0.9995175	0.999789
R^2, FR Training				
Regressors/Neurons	6	8	10	12
2	0.5205786	0.6925345	0.4461068	0.5551839
3	0.6705247	0.8419732	0.9252495	0.9243027
4	-0.4137852	0.7838424	0.813022	0.8983720
5	-0.972349	-0.6521976	-0.1789108	0.6314544
R^2, OSA Test				
Regressors/Neurons	6	8	10	12
2	0.9996252	0.9996924	0.9997235	0.9997428
3	0.999305	0.9995413	0.9996834	0.9997292
4	0.9993952	0.9994318	0.9997608	0.9996364
5	0.9989228	0.9995542	0.9995175	0.9997882
R^2, FR Test				
Regressors/Neurons	6	8	10	12
2	0.5193330	0.6725372	0.4456376	0.5560610
3	0.7045499	0.8524704	0.9278230	0.9266442
4	-0.4102181	0.7823735	0.7920774	0.8976120
5	-0.9710301	-0.6395710	-0.1823629	0.6476350

Table 3.2: Multiple Correlation Coefficients for Batch 128 and Learning Rate 0.001 (1 DoF).

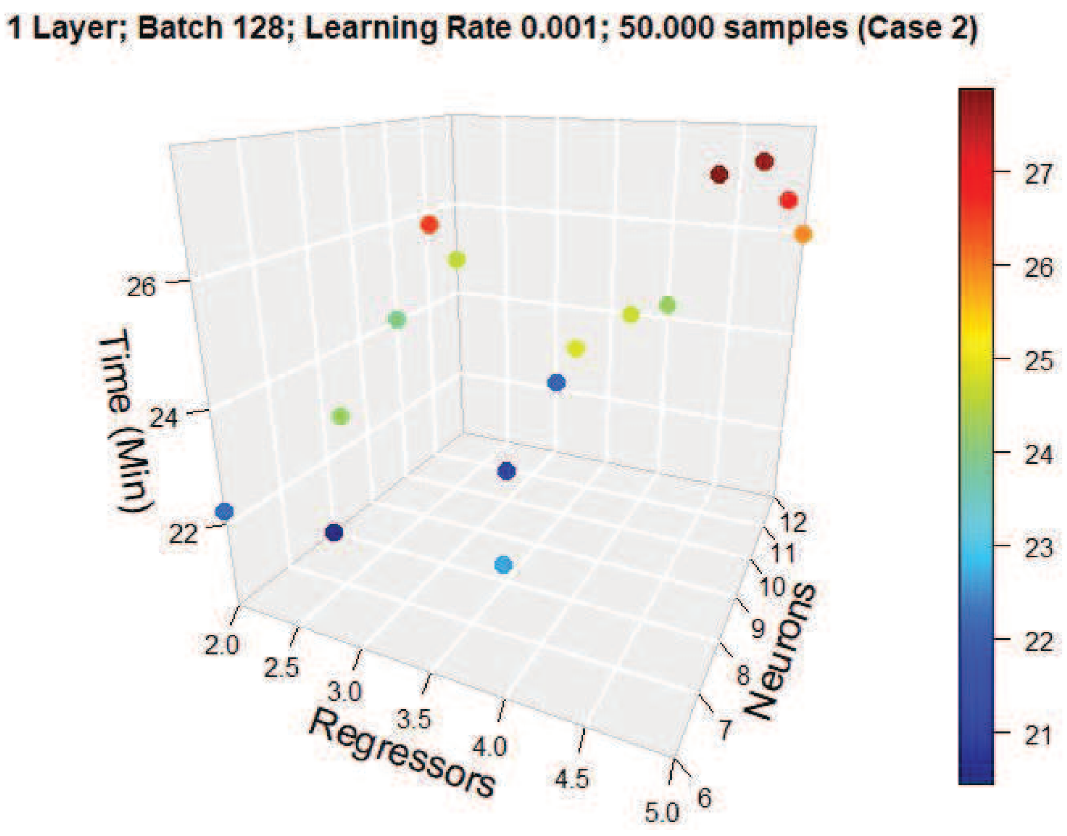

Figure 3.3: 3D Graph showing how long take to build the model with the combination of regressors, neurons and parameters.

For the case 2, it is possible to see that all R^2 values of OSA for training and test are greater than 0.9 in any combination of regressors and neurons. Otherwise the R^2 for FR in training and test got some negative values. These negative values were concentrated in a region which has a great number of regressors and a small number of neurons, probably indicating that increase number of regressors and diminishing number of neurons is not a good option. While the values of R^2 in FR greater than 0.9 got 3 regressors and the two largest numbers of neurons (10 and 12). Note that the average time to compile this package of models was 24 minutes, being smaller than the average time of case 1. This happens because the increment of batch from 1 in case 1 to 128 in case 2, that shows batch has a strong influence on the build time of the model.

In the graph shown there is much fluctuation of the combinations between neurons and regressors in relation to the time of construction of the model. However is possible to see that with less neurons the elapsed time to build the model is lower.

1 Layer; Batch 1; Learning Rate 0.01; 50.000 Samples (Case 3)				
Average Time 1h 43 min				
R^2, OSA Training				
Regressors/Neurons	6	8	10	12
2	0.9932163	0.9958785	0.9945194	0.9909835
3	0.9773624	0.9804612	0.9879727	0.9788836
4	0.9571679	0.9792299	0.9235091	0.9589338
5	0.8456940	0.9069847	0.9051564	0.9090456
R^2, FR Training				
Regressors/Neurons	6	8	10	12
2	0.4034994	0.9033679	0.2277386	0.3668113
3	-0.1460767	-0.6094115	-0.1706908	-0.3479258
4	0.0105672	-0.0698936	-0.9058537	-1.5241200
5	-1.1949890	-1.0402710	-0.9074499	-1.5773810
R^2, OSA Test				
Regressors/Neurons	6	8	10	12
2	0.9933322	0.9957371	0.9945502	0.9909655
3	0.9769320	0.9806352	0.9879411	0.9785662
4	0.9571873	0.9795893	0.9236016	0.9584784
5	0.8444545	0.9080350	0.9027419	0.9093385
R^2, FR Test				
Regressors/Neurons	6	8	10	12
2	0.4018950	0.9033505	0.2253305	0.3647295
3	-0.1457781	-0.6097267	-0.1701890	-0.3469355
4	0.0106835	-0.0695541	-0.9052779	-1.5223450
5	-1.1941800	-1.0389940	-0.9059696	-1.5746660

Table 3.3: Multiple Correlation Coefficients for Batch 1 and Learning Rate 0.01 (1 DoF).

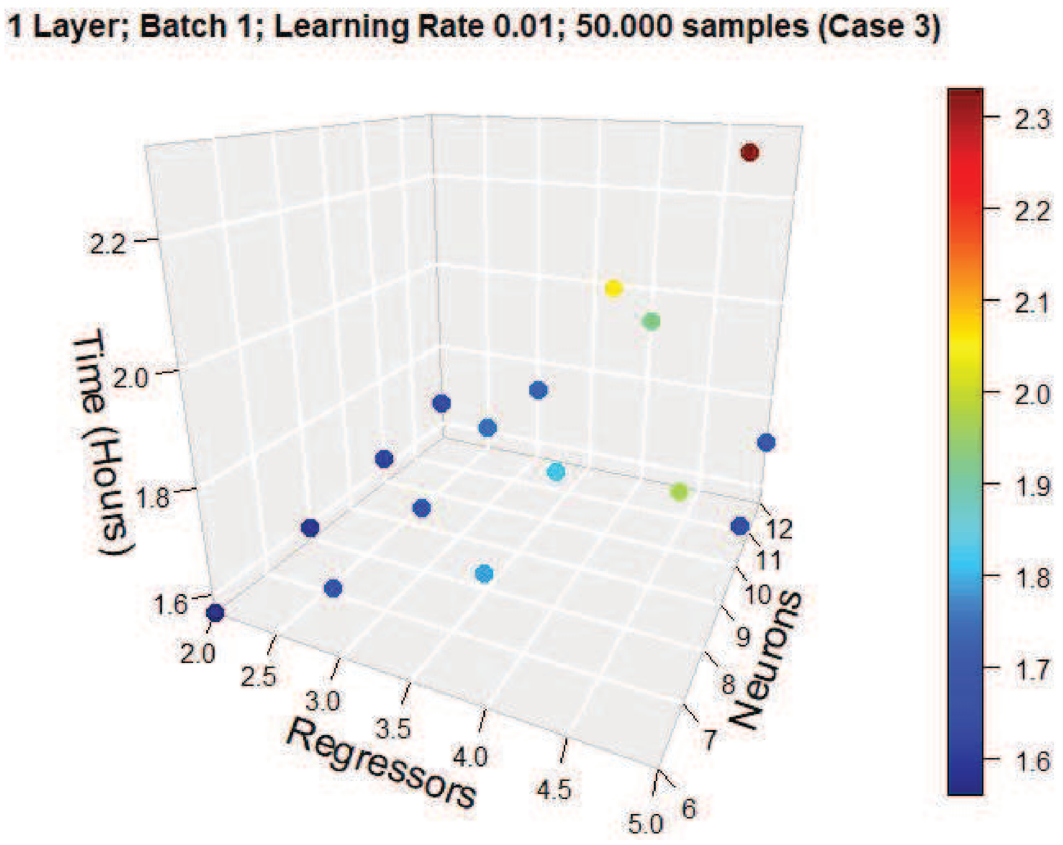

Figure 3.4: 3D Graph showing how long take to build the model with the combination of regressors, neurons and parameters.

For the case 3, it is possible to see that only one model has values of R^2 lower than 0.9 for OSA in training and test, however greater than 0.8. Otherwise the R^2 for FR in training and test obtained, in majority, negative values. Only for 2 regressors all the values are positives combining with any of the numbers of neurons and only one model had values of training and test greater than 0.9: 2 regressors and 8 neurons. Note that the average time to compile this package of models was 1 hour and 43 minutes same as the case 1, which has the same batch size, showing once more the strong influence of batch has on the build time of the model.

In the graph is showed that with less neurons and regressors the elapsed time to build the model is lower with exception of 3 points out of the curve.

1 Layer; Batch 128; Learning Rate 0.01; 50.000 Samples (Case 4)				
Average Time 26 min				
R^2, OSA Training				
Regressors/Neurons	6	8	10	12
2	0.9963244	0.9991037	0.9991008	0.9985764
3	0.9972108	0.9980686	0.9980096	0.9984052
4	0.9976949	0.9919646	0.9971071	0.9979620
5	0.9954367	0.9953074	0.9941696	0.9957953
R^2, FR Training				
Regressors/Neurons	6	8	10	12
2	0.8133385	0.7805277	0.8233956	0.6931343
3	0.8306456	0.9719331	0.9308468	0.9688430
4	0.7006301	0.9046013	0.9733141	0.9841038
5	0.7143235	0.9388632	0.4770590	-0.0280644
R^2, OSA Test				
Regressors/Neurons	6	8	10	12
2	0.9963029	0.9990892	0.9991418	0.9985178
3	0.9972184	0.9980824	0.9980341	0.9984284
4	0.9976883	0.9919548	0.9971134	0.9979664
5	0.9954336	0.9953436	0.9942626	0.9957359
R^2, FR Test				
Regressors/Neurons	6	8	10	12
2	0.8129247	0.7816866	0.8223943	0.6917121
3	0.8311256	0.9719060	0.9307424	0.9687896
4	0.7014565	0.9048482	0.9736321	0.9841195
5	0.7148792	0.9390010	0.4774554	-0.0309065

Table 3.4: Multiple Correlation Coefficients for Batch 128 and Learning Rate 0.01 (1 DoF).

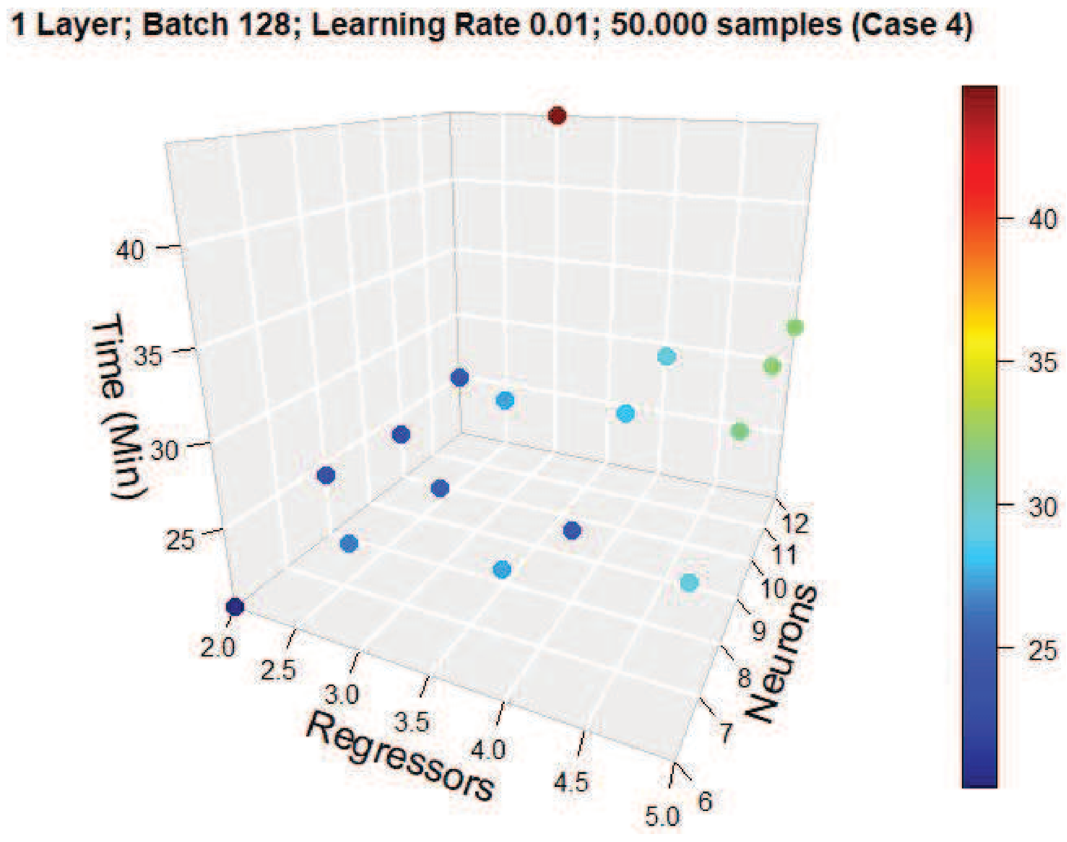

Figure 3.5: 3D Graph showing how long take to build the model with the combination of regressors, neurons and parameters.

For the case 4, it is possible to see that all R^2 values of OSA for training and test are greater than 0.9 in any combination of regressors and neurons. Otherwise the R^2 for FR in training and test got some values less than 0.9 and even one negative value. While the values of R^2 in FR greater than 0.9 are concentrated in majority in the line of 3 and 4 regressors for 8, 10 and 12 neurons. Is not necessary look for the combination with more neurons once that doing it the complexity will increase and the time to build the model will also increase. Note that the average time to compile this package of models was 26 minutes, very close of case 2, which has the same batch size, showing once more the strong influence of batch has on the build time of the model.

In the graph is showed that, in general, with less neurons and regressors the elapsed time to build the model is lower with exception of 1 point out of the curve.

In general OSA has values greater than 0.9, this can be an effect of behavioral pattern in the sample or due to the large sampling used to train the model. Otherwise the FR in relation to OSA, obtained less values greater than 0.9, this is due to the parameters which compose the FR method which accumulates the residuals after each iteration. It can be noticed in the tables that the worst combinations, in general, are made with few neurons and many regressors. On the other hand, the best combinations, in general, are made with many neurons and few regressors.

3.2 Selection of the 1 DoF model

For the selection of the best model, in table 3.1 was compiled all the values obtained for FR test greater than 0.9 in order to compare them.

Model Number	R^2, FR Test	Regressors	Neurons	Elapsed Time (min)	Batch	Learning Rate	Layer
1	0.9972837	2	8	91	1	0.001	1
2	0.9968095	2	10	95	1	0.001	1
3	0.9997272	2	12	96	1	0.001	1
4	0.9995326	3	12	101	1	0.001	1
5	0.9319252	4	12	154	1	0.001	1
6	0.9298589	4	12	24	128	0.001	1
7	0.9033505	2	8	95	1	0.01	1
8	0.9719060	3	8	25	128	0.01	1
9	0.9307424	3	10	28	128	0.01	1
10	0.9687896	3	12	45	128	0.01	1
11	0.9048482	4	8	25	128	0.01	1
12	0.9736321	4	10	28	128	0.01	1
13	0.9841195	4	12	29	128	0.01	1
14	0.9390010	5	8	32	128	0.01	1

Table 3.5: Compilation of the best models in relation to R^2 FR Test (1 DoF).

Based in comparisons between all the models showed in table 3.5 and looking for the simplest models with good accuracy results and small elapsed time, the model chosen was number 12, which has batch 128, learning rate 0.01 (case 4), 4 regressors and 10 neurons, which takes 28 minutes to build the model and has training for OSA equal to 0.9971071, training for FR equal to

0.9733141, test for OSA equal to 0.9971134 and test for FR equal to 0.9736321 i.e. all values of R^2 are greater than 0.97. This model was chosen also because it gives a stable model, i.e, it was rebuild some times and the model don't have a huge change in your correlation coefficients after each training.

Beside of the model chosen, if necessary to get better results without carry with the time elapsed in consideration, it is possible to chose the model number 1 which has a R^2 of 0.9972837 and will provide a better accuracy.

3.3 Results for 2 DoF

The piezoelectric cantilever in micromanipulators with 2 DoF, different of 1 DoF, has drawback of coupling effect. This effect make the cantilever under voltage move in both directions x and y (Figure 2.7). Then 2 outputs are produced for x and y directions. For 2 DoF were built 2 models, each model were built for each output and receives samples of 2 inputs and 1 output (for training and test). The NARX method with ANNs were used by varying the order of the regressors in both inputs and the output (all with the same order always) in order to check how the accuracy of the model changes by gradually increasing the number of regressors and neurons.

For the first output was defined the number of regressors ranging from 4 to 9 and making a combination between the regressors and the number of neurons, which are ranging from 145 to 205 with step 15, thus creating 30 different models. It is important to note that all models made here were made with only 1 batch, 1 deep layer and 100 epochs. Each combination of regressors and neurons was made for three learning rates: 0.01, 0.001 and 0.0001. In this way, was possible to obtain 90 models for the fist output (figure 3.6).

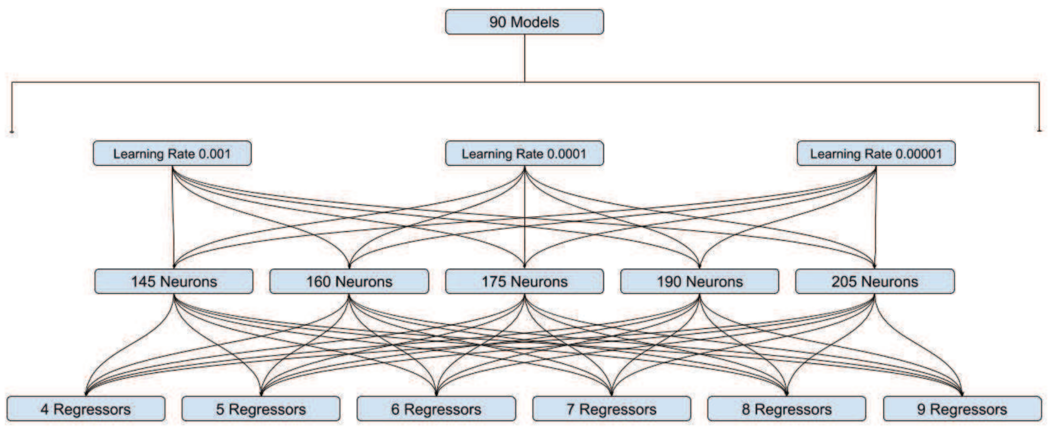

Figure 3.6: Combination of parameters used in this document to form 54 models used in 2-DoF problem.

For the first output models, we saved the following correlation coefficients R^2: for One-step-ahead (OSA) and Free-run (FR), both for validation and training phases, in order to compare all of them and choose the best result of model. Here were not saved the time elapsed to build the model, because in general the time elapsed to build the 2 DoF models are almost the same (around 3 min each model) and is faster than the time elapsed to build 1 DoF models, due to the bigger number of samples that 1 DoF has. All the models were put in tables and the tables were divided in cases for each learning rate chosen.

1 Layer; Batch 1; Learning Rate 0.001; 500 Samples (Case 1)						
R^2, OSA Training						
Neurons/Regressors	4	5	6	7	8	9
145	0.9986614	0.9975264	0.9983008	0.997356	0.9965636	0.9985976
160	0.9977542	0.9987513	0.9984684	0.9982018	0.9978504	0.9981393
175	0.9987481	0.9984042	0.9983196	0.9982075	0.9985128	0.9979437
190	0.9987102	0.9988671	0.9979753	0.9983986	0.998223	0.9981071
205	0.9990119	0.9987094	0.9978408	0.9984953	0.9986279	0.9985423
R^2, FR Training						
Neurons/Regressors	4	5	6	7	8	9
145	0.862347	0.04929068	0.9894701	0.00233991	0.7580069	0.9010972
160	-0.2608189	0.9451563	0.1834611	0.6915893	0.8209727	0.9864567
175	0.9959253	0.6794783	0.1080795	0.9869321	0.9958667	0.9449763
190	0.9864332	0.9950858	0.03256669	0.9880586	0.9943644	0.9929051
205	0.9949568	0.6998308	0.9925817	0.9887439	0.9075991	0.9924626
R^2, OSA Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.9961759	0.9962852	0.9946631	0.9964043	0.9930894	0.9970135
160	0.9966247	0.9968689	0.9967062	0.995649	0.9952625	0.9961886
175	0.9958961	0.9961031	0.9956776	0.9959496	0.996761	0.9962544
190	0.9963376	0.9970716	0.9966082	0.9963223	0.9964988	0.9973897
205	0.9967842	0.9969001	0.9965406	0.9965745	0.9967946	0.9960455
R^2, FR Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.6575496	0.0556689	0.9663191	0.0099241	0.6619129	0.9931115
160	-0.0410240	0.8359197	0.2514659	0.6678359	0.7406800	0.8504922
175	0.9580344	0.6363985	0.1172827	0.9705652	0.8555071	0.8529002
190	0.7923266	0.9902354	0.1792099	0.7543438	0.9806411	0.9864209
205	0.9928115	0.8019694	0.9880154	0.9843496	0.7893304	0.9842715

Table 3.6: Multiple Correlation Coefficients for Learning Rate 0.001 (2 DoF - 1ST output).

1 Layer; Batch 1; Learning Rate 0.0001; 500 Samples (Case 2)						
R^2, OSA Training						
Neurons/Regressors	4	5	6	7	8	9
145	0.9982662	0.9984519	0.9986129	0.9987748	0.9992084	0.9982692
160	0.9984468	0.9986105	0.9991896	0.9989955	0.9994156	0.9990378
175	0.9987983	0.9989337	0.9989813	0.9992974	0.99931	0.9992254
190	0.9986917	0.9991036	0.9992496	0.999375	0.9992326	0.9993627
205	0.9988739	0.9992845	0.9993078	0.9995117	0.999518	0.9994933
R^2, FR Training						
Neurons/Regressors	4	5	6	7	8	9
145	0.9922119	0.9935742	0.09392488	0.9925617	0.9898151	0.986024
160	0.9616721	0.9875445	0.9856295	0.4982136	0.523414	0.8186685
175	0.08808451	0.9921332	0.2928431	0.9899251	0.9908928	0.9680588
190	0.1072489	0.9902111	0.2601345	0.379472	0.9885377	0.6484878
205	0.4757388	0.5081645	0.9921495	0.9908208	0.8259167	0.9927662
R^2, OSA Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.9952588	0.9947449	0.994916	0.9952934	0.9961952	0.9945395
160	0.9955483	0.9956068	0.9965505	0.995513	0.9969239	0.9956244
175	0.9965387	0.9964881	0.9960841	0.9961365	0.9965266	0.9966629
190	0.9968549	0.9966868	0.9964881	0.9969394	0.9954459	0.9967107
205	0.9965166	0.9970699	0.9966107	0.9971896	0.9970402	0.997082
R^2, FR Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.9797028	0.979581	-0.0039014	0.9869789	0.9856614	0.982686
160	0.906069	0.9012067	0.9632767	0.1038866	0.6906955	0.9820897
175	0.8635429	0.9585072	0.1821907	0.9833816	0.9863514	0.9780014
190	0.4376468	0.9767255	0.3510986	0.6810725	0.9740704	0.8788341
205	0.0798337	0.4656284	0.9157195	0.9768344	0.9425035	0.9882593

Table 3.7: Multiple Correlation Coefficients for Learning Rate 0.0001 (2 DoF - 1ST output).

1 Layer; Batch 1; Learning Rate 0.00001; 500 Samples (Case 3)						
R^2, OSA Training						
Neurons/Regressors	4	5	6	7	8	9
145	0.5986768	0.6457335	0.6083497	0.6138595	0.6363258	0.6085248
160	0.6747787	0.6804224	0.6488642	0.657246	0.6237461	0.6736876
175	0.6853711	0.7223644	0.7192783	0.7053398	0.6932741	0.6966688
190	0.7770856	0.7423055	0.742892	0.7436222	0.7074937	0.7335795
205	0.8116791	0.754495	0.7958073	0.7565614	0.7574878	0.7618261
R^2, FR Training						
Neurons/Regressors	4	5	6	7	8	9
145	0.1031794	-0.0485545	-0.2390091	-0.0827425	-0.2873559	0.2667048
160	0.6473319	-0.1281776	-0.04151905	-0.1844581	-0.0693717	0.6642077
175	0.06824042	-0.0010346	-0.2129984	-0.2046245	-0.4529675	0.6875974
190	-0.1931267	-0.386371	-0.1067817	-0.3994835	-0.2009121	0.2729826
205	-0.0012419	-0.151641	-0.388255	-0.5272722	-0.4012677	0.2362106
R^2, OSA Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.5652327	0.6075328	0.5685361	0.5679355	0.5940797	0.5687744
160	0.6397609	0.6439763	0.6046997	0.6141026	0.5786678	0.6342203
175	0.6520428	0.6825876	0.6761558	0.6634414	0.6482098	0.6553333
190	0.7450106	0.705697	0.6994855	0.6980191	0.6654916	0.6921287
205	0.7809434	0.7156713	0.7539262	0.7062678	0.7150601	0.7217461
R^2, FR Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.0927180	0.0402869	-0.2644856	-0.1266243	-0.2914266	0.3767803
160	0.5619591	-0.1770918	0.0896918	-0.220421	-0.116001	0.6264218
175	0.04052363	0.1216265	-0.2842086	-0.2519529	-0.4137436	0.6468319
190	-0.1406955	-0.3634035	0.1031531	-0.4186212	-0.0923154	0.4196988
205	0.0260836	-0.0624941	-0.163769	-0.4466456	-0.2711802	0.3816805

Table 3.8: Multiple Correlation Coefficients for Learning Rate 0.00001 (2 DoF - 1ST output).

For all the tables shown, it is possible to find just three models with FR test greater than 0.99. For this case was choose the model with 1 layer, batch 1, learning rate 0.001, 205 neurons and 4 regressors (Case 1) having OSA training 0.9990119, FR training 0.9949568, OSA test 0.9967842 and FR test 0.9928115. This model was choose because of your R^2 greater than 0.99, your model is less complex than the two others with R^2 greater than 0.99 and it gives the best graph of correlation coefficient. Figure 3.7 shows the correlation coefficient R^2 for OSA and FR.

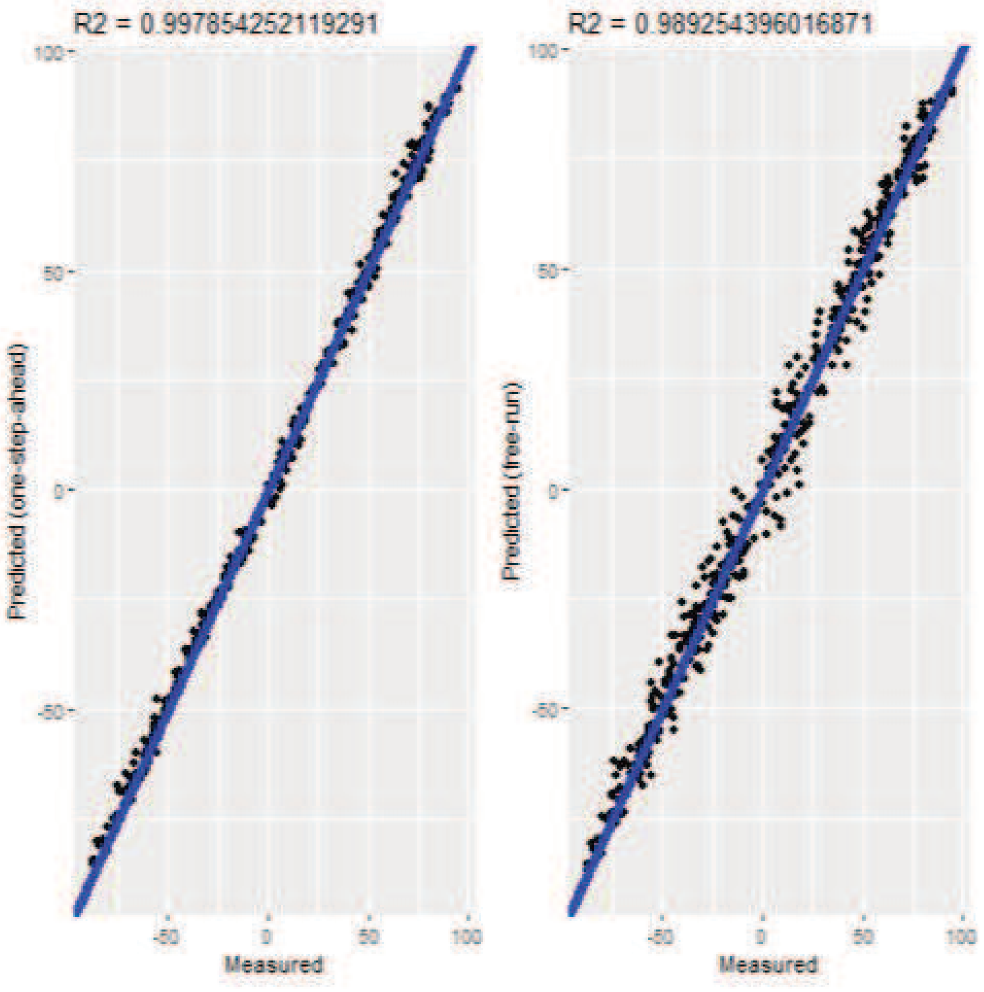

Figure 3.7: Correlation Coefficient of One-Step-Ahead and Free-Run for 2 DoF (1ST output).

For the second output was also defined the number of regressors ranging from 4 to 9 and making a combination between the regressors and the number of neurons, which are ranging from 145 to 205 with step 15, thus creating 30 different models. The number of neurons for this output increased largely because the samples don't have an easy pattern to be recognized by few neurons, as tested. Then were chosen more neurons which could give some R^2 values reasonable. It is important to note that all models made here were made with only 1 batch, 1 deep layer and 100 epochs. Each combination of regressors and neurons was made for three learning rates: 0.001, 0.0001 and 0.00001. In this way, was possible to obtain 90 models for the second output (figure 3.8).

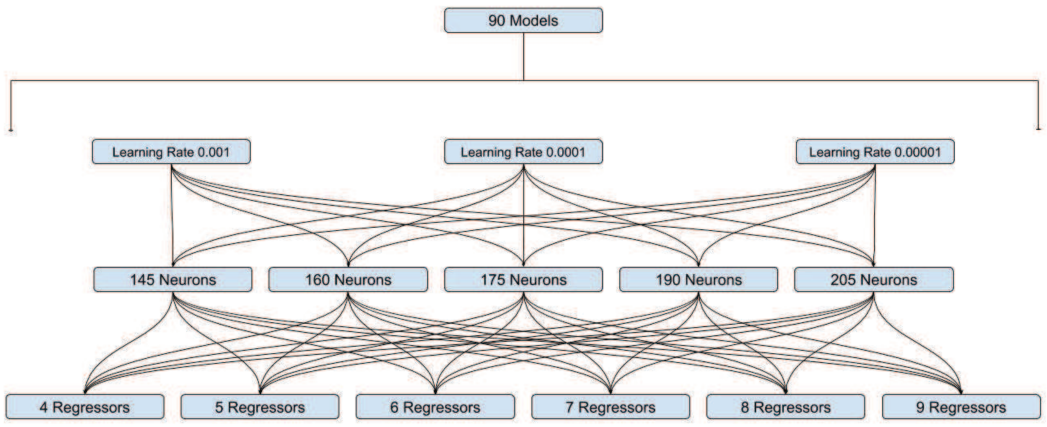

Figure 3.8: Combination of parameters used in this document to form 90 models used in 2-DoF problem.

For the second output models, we saved the following correlation coefficients R^2: for One-step-ahead (OSA) and Free-run (FR), both for validation and training phases, in order to compare all of them and choose the best result of model. Here were not saved also the time elapsed to build the model, because in general the time elapsed to build the 2 DoF models are almost the same (around 3 min each model) and is faster than the time elapsed to build 1 DoF models, due to the bigger number of samples that 1 DoF has. All the models were put in tables and the tables were divided in cases by the learning rate chosen.

1 Layer; Batch 1; Learning Rate 0.001; 500 Samples (Case 1)						
R^2, OSA Training						
Neurons/Regressors	4	5	6	7	8	9
145	0.984420	0.996188	0.988082	0.988700	0.979877	0.977979
160	0.991419	0.990801	0.990780	0.991054	0.983507	0.985242
175	0.982497	0.989078	0.988999	0.988707	0.985031	0.988099
190	0.989415	0.995728	0.992251	0.993071	0.988363	0.980504
205	0.989665	0.995206	0.984330	0.986363	0.989286	0.981461
R^2, FR Training						
Neurons/Regressors	4	5	6	7	8	9
145	0.810541	0.982745	0.966348	0.939256	0.930804	0.923943
160	0.741600	0.837370	0.962720	0.974709	0.933314	0.974685
175	0.822956	0.966982	0.900249	0.970214	0.928516	0.963051
190	0.855075	0.990889	0.984850	0.976502	0.967932	0.919951
205	0.908366	0.979607	0.888652	0.953672	0.943226	0.939270
R^2, OSA Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.981512	0.988858	0.988552	0.984228	0.975913	0.969064
160	0.982107	0.986570	0.979415	0.980556	0.962322	0.973791
175	0.969257	0.986259	0.986663	0.982281	0.974789	0.981989
190	0.990279	0.986801	0.986626	0.982547	0.983049	0.969195
205	0.987285	0.990428	0.973793	0.982297	0.978741	0.974171
R^2, FR Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.514853	0.755786	0.779658	0.735141	0.610970	0.599902
160	0.553455	0.734377	0.728604	0.807018	0.655752	0.719300
175	0.438340	0.718250	0.695538	0.769465	0.681276	0.753310
190	0.701416	0.764030	0.787875	0.772075	0.749681	0.660933
205	0.718226	0.767689	0.663842	0.773372	0.656809	0.728469

Table 3.9: Multiple Correlation Coefficients for Learning Rate 0.001 (2 DoF - 2ND output).

1 Layer; Batch 1; Learning Rate 0.0001; 500 Samples (Case 2)						
R^2, OSA Training						
Neurons/Regressors	4	5	6	7	8	9
145	0.996383	0.987201	0.962569	0.931673	0.930098	0.735507
160	0.998269	0.992395	0.942272	0.947543	0.848237	0.876533
175	0.998231	0.994514	0.978529	0.942400	0.928188	0.811701
190	0.998435	0.997294	0.986529	0.946772	0.947413	0.955672
205	0.998739	0.998142	0.994988	0.983410	0.896736	0.923407
R^2, FR Training						
Neurons/Regressors	4	5	6	7	8	9
145	0,684823	0,942540	0,558101	0,364297	0,530595	-0,09300
160	0,497286	0,948144	0,694513	0,564760	-0,32520	-0,03832
175	0,477881	0,958926	0,916339	0,281005	-0,29712	-0,05827
190	0,585022	0,986901	0,912671	0,687224	0,250800	0,715680
205	0,937739	0,994405	0,980129	0,908921	0,232431	0,315894
R^2, OSA Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.986553	0.962959	0.927439	0.881446	0.877375	0.649143
160	0.995175	0.971153	0.900118	0.906835	0.767244	0.801558
175	0.993695	0.973696	0.948566	0.892750	0.879990	0.734316
190	0.994417	0.985387	0.958842	0.912532	0.902196	0.911710
205	0.994807	0.988460	0.974868	0.951408	0.839987	0.869662
R^2, FR Test						
Neurons/Regressors	4	5	6	7	8	9
145	0.467598	0.725513	0.478785	0.194026	0.37330	-0.25491
160	0.261436	0.736683	0.461292	0.454476	-0.40818	-0.19540
175	0.377208	0.743502	0.713766	0.099949	-0.38459	-0.25477
190	0.421438	0.762865	0.719154	0.503250	-0.03470	0.542912
205	0.688181	0.779923	0.765361	0.507194	-0.03425	0.063024

Table 3.10: Multiple Correlation Coefficients for Learning Rate 0.0001 (2 DoF - 2ND output).

1 Layer; Batch 1; Learning Rate 0.00001; 500 Samples (Case 3)						
R^2, OSA Training						
Neurons/Regressors	4	5	6	7	8	9
145	-2.39551	-2.66925	-2.75223	-2.77076	-2.87466	-2.94477
160	-2.10997	-1.93532	-1.81270	-2.18956	-2.29396	-2.18914
175	-1.58955	-1.64350	-1.88783	-1.79685	-1.76635	-2.23881
190	-1.41554	-1.43490	-1.36475	-1.28076	-1.38429	-1.54803
205	-1.06870	-1.23436	-1.15413	-1.12941	-1.18107	-1.21073
R^2, FR Training						
Neurons/Regressors	4	5	6	7	8	9
145	-2.38968	-2.65312	-2.7451	-2.75339	-2.83187	-2.91529
160	-2.10345	-1.93973	-1.80866	-2.17639	-2.26613	-2.17362
175	-1.60020	-1.64150	-1.87886	-1.78288	-1.74423	-2.22205
190	-1.42362	-1.44043	-1.36715	-1.27896	-1.38276	-1.54637
205	-1.09669	-1.24272	-1.13724	-1.13748	-1.17522	-1.20863
R^2, OSA Test						
Neurons/Regressors	4	5	6	7	8	9
145	-1.36227	-1.58680	-1.64745	-1.65536	-1.76340	-1.79025
160	-1.15509	-1.01760	-0.93974	-1.22096	-1.30274	-1.21771
175	-0.76411	-0.81831	-0.99250	-0.94204	-0.92155	-1.26074
190	-0.64965	-0.66328	-0.62625	-0.56640	-0.65069	-0.75717
205	-0.39161	-0.52929	-0.50239	-0.45974	-0.51189	-0.51925
R^2, FR Test						
Neurons/Regressors	4	5	6	7	8	9
145	-1.35407	-1.55278	-1.62226	-1.62402	-1.68217	-1.74564
160	-1.14458	-1.02753	-0.93461	-1.19620	-1.25825	-1.19059
175	-0.78723	-0.81729	-0.98470	-0.91431	-0.88480	-1.22633
190	-0.66715	-0.67961	-0.63064	-0.56880	-0.63643	-0.74707
205	-0.45226	-0.54806	-0.48012	-0.47794	-0.49984	-0.52159

Table 3.11: Multiple Correlation Coefficients for Learning Rate 0.00001 (2 DoF - 2ND output).

Note that in the tables 3.6 to 3.11 it was impossible to find any value of R^2 greater than 0.9. It is due to the patterns of the samples and the small number of samples used to train the model. It is possible to note also the learning rate of 0.00001 provides only negative values for R^2. Probably the low learning rate made the gradient descent stuck in a bad value.

For all the tables shown, it is possible to find just one model with FR test greater than 0.8. This model has 1 layer, batch 1, learning rate 0.001, 160 neurons and 7 regressors (Case 1) having OSA training 0.991054, FR training 0.974709, OSA test 0.980556 and FR test 0.807018. Figure 3.9 shows the correlation coefficient R^2 for OSA and FR.

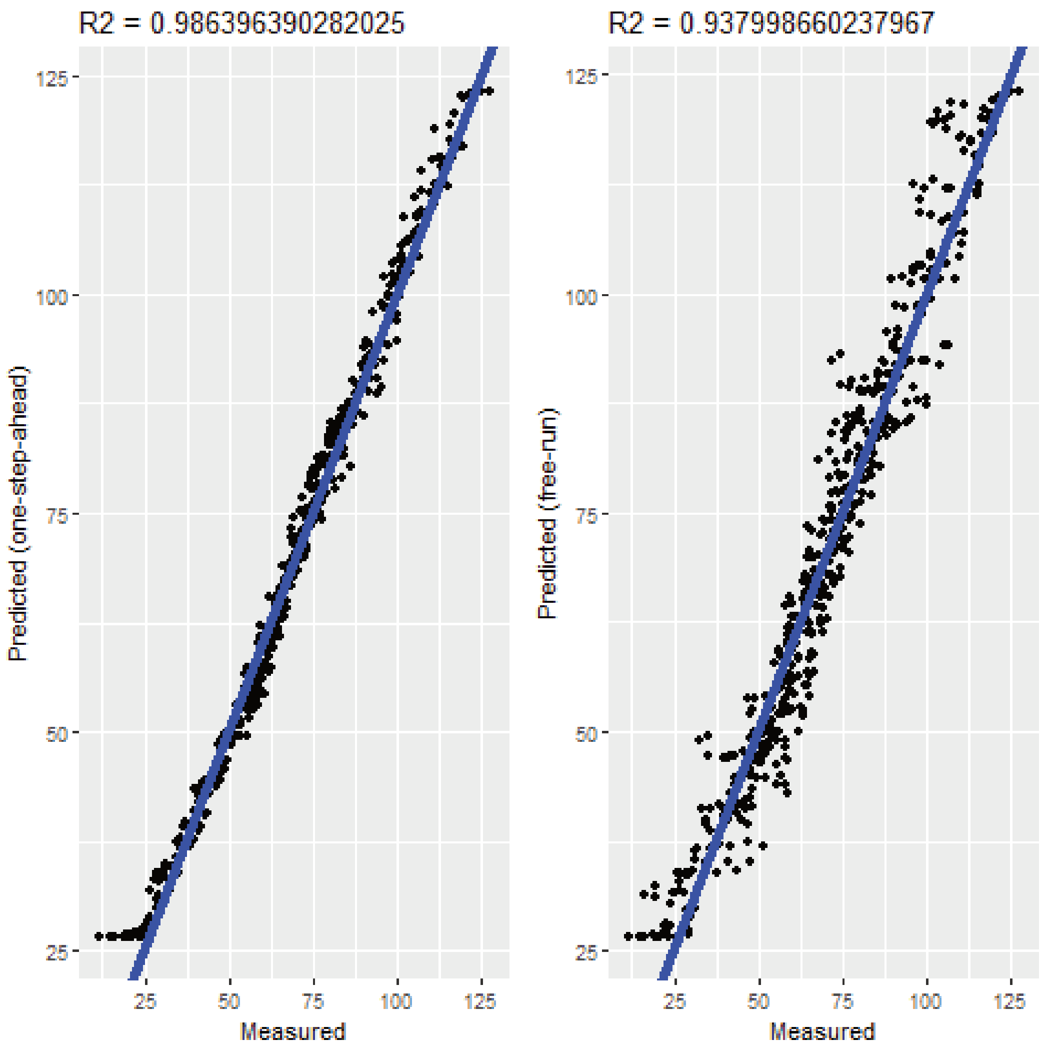

Figure 3.9: Correlation Coefficient of One-Step-Ahead and Free-Run for 2 DoF (2ND output).

4 CONCLUSIONS

The chose of the model for 1 DoF has shown balance between the time elapsed to build the model through training and a high accuracy given by the R^2 greater than 0.9. While the chose of the model for 2 DoF was divided in 2 models. The first model used for the first output showed a relevant result with R^2 greater than 0.9. However for the second model, used to model the second output, it was not possible to obtain a value for R^2 in Free-run test greater than 0.9, but greater than 0.8. This is because of the small number of samples. Note that for 2 DoF the time to build models was faster than 1 DoF, because while 1 DoF has 50000 samples, the 2 DoF samples were only 500, however the complexity was greater once that both models of 2 DoF used more than 100 neurons.

The present document resulted in models that use system identification with neural network to provide feedback for piezoelectric micromanipulators, making the use of piezoelectric micromanipulators more precise and helping to solve tasks more complex.

Beside of the large number of models obtained in this document, there are infinite possibilities of combination of regressors, neurons, layers, batches and learning rates not tested here. Then for future research the search for new models using these parameters could be done and even using others optimization algorithms.

Bibliography

Aljanaideh, O. & Rakotondrabe, M. (2018). Observer and robust H_∞ control of a 2-dof piezoelectric actuator equipped with self-measurement. *IEEE Robotics and Automation Letters*, *3*(2), 1080–1087.

Ayala, H. V. H., Habineza, D., Rakotondrabe, M., Klein, C. E. & Coelho, L. S. (2015). Nonlinear black-box system identification through neural networks of a hysteretic piezoelectric robotic micromanipulator. *IFAC-PapersOnLine*, *48*(28), 409–414.

Ayala, H. V. H., Rakotondrahe, M. & Coelho, L. (2018). Modeling of a 2-dof piezoelectric micromanipulator at high frequency rates through nonlinear black-box system identification. In *2018 annual american control conference (acc)* (pp. 4354–4359).

Chen, S., Billings, S. & Grant, P. (1990). Non-linear system identification using neural networks. *International journal of control*, *51*(6), 1191–1214.

Chollet, F. & Allaire, J. J. (2018). *Deep learning with r*. Manning Publications Company.

Cohen, P. R. & Feigenbaum, E. A. (2014). *The handbook of artificial intelligence* (Vol. 3). Butterworth-Heinemann.

De la Rosa, E. & Yu, W. (2016). Randomized algorithms for nonlinear system identification with deep learning modification. *Information Sciences*, *364*, 197–212.

Genc, S. (2017). Parametric system identification using deep convolutional neural networks. In *Neural networks (ijcnn), 2017 international joint conference on* (pp. 2112–2119).

Goldfarb, M. & Celanovic, N. (1997). Modeling piezoelectric stack actuators for control of micromanipulation. *IEEE control systems*, *17*(3), 69–79.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. (2016). *Deep learning* (Vol. 1). MIT press Cambridge.

Habineza, D., Rakotondrabe, M. & Le Gorrec, Y. (2015). Bouc-Wen modeling and feedforward control of multivariable hysteresis in piezoelectric systems: application to a 3-dof piezotube scanner. *IEEE Transactions on Control Systems Technology*, *23*(5), 1797–1806.

Haykin. (2009). *Neural networks and learning machines* (Vol. 3). Pearson Upper Saddle River, NJ, USA:.

Karlik, B. & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, *1*(4), 111–122.

Knerr, S., Personnaz, L. & Dreyfus, G. (1990). Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing* (pp. 41–50). Springer.

Nakayama, T., Fujiwara, H., Yamada, S., Tastumi, K., Honda, T. & Fujii, S. (1999). Clinical application of a new assisted hatching method using a piezo-micromanipulator for morphologically low-quality embryos in poor-prognosis infertile patients. *Fertility and Sterility*, *71*(6), 1014–1018.

Noël, J.-P., Esfahani, A. F., Kerschen, G. & Schoukens, J. (2017). A nonlinear state-space approach to hysteresis identification. *Mechanical Systems and Signal Processing*, *84*, 171–184.

Oh, K.-S. & Jung, K. (2004). Gpu implementation of neural networks. *Pattern Recognition*, *37*(6), 1311–1314.

Rakotondrabe, M. (2013). *Smart materials-based actuators at the micro/nano-scale: Characterization, control, and applications.* Springer Science & Business Media.

Rakotondrabe, M. (2017). Multivariable classical prandtl–ishlinskii hysteresis modeling and compensation and sensorless control of a nonlinear 2-dof piezoactuator. *Nonlinear Dynamics*, *89*(1), 481–499.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117.

Tan, K. & Ng, S. (2001). Computer controlled piezo micromanipulation system for biomedical applications. *Engineering Science and Education Journal*, *10*(6), 249–256.

Zhang, Y. L., Han, M. L., Yu, M. Y., Shee, C. Y. & Ang, W. T. (2012). Automatic hysteresis modeling of piezoelectric micromanipulator in vision-guided micromanipulation systems. *IEEE/ASME Transactions on Mechatronics*, *17*(3), 547–553.

Žilková, J., Timko, J. & Girovský, P. (2006). Nonlinear system control using neural networks. *Acta Polytechnica Hungarica*, *3*(4), 85–94.