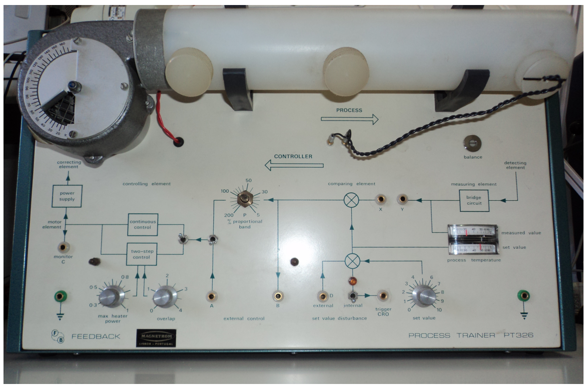
INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa



Multiple Model Identification in Systems with Variable Dynamics

Tiago Alexandre de Almeida Simão

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Júri

Presidente:	Doutor Carlos Filipe Gomes Bispo
Orientador:	Doutor João Manuel Lage de Miranda Lemos
Vogal:	Doutor Jorge dos Santos Salvador Marques

Outubro de 2011

Acknowledgments

First of all, I would like to thanks to my supervisor Professor João Miranda Lemos for the support and tips he provided during my thesis and the patience he had in reading all my reports and dissertation. I also want to give a word of appreciation to FCT for the research grant attributed to me.

I cannot forget to mention my gratitude to my colleges and my friends who accompanied and helped me since I meet them.

To my family, I thank all the unconditional and inestimable support, and also appreciate all the knowledge I could not learn from anyone else.

This work has been performed in the framework of project ARGUS - Activity recognition and object tracking based on multiple models, PTDC/EEA-CRO/098550/2008.

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

Abstract

In this dissertation, we present an example of an application to control a nonlinear system using the multiple models approach. The system has variable dynamics, where a model can represent each dynamic. For each model a controller is designed. As the system dynamics is externally manipulated, and therefore it cannot be directly measured, we use of a supervisor to detect the dynamic changes.

To control a nonlinear system, we first must identify some characteristics that enable us to create a model. As a nonlinear system has various incremental dynamics depending on the operating point, we can identify a set of local models, where each model corresponds to a specific operating point. With the multiple models obtained from system identification, we can design a controller formed by the patching of local controllers, each one designed for a corresponding local model. The supervisor detects which model is similar to the behavior of the current system and chooses the corresponding controller. In order to ensure stability, a dwell time switching logic is used. The model offsets are also estimated during the experiments. An original supervisor algorithm that tackles the presence of unknown offsets is purposed and demonstrated experimentally.

After the controllers and supervisor design, we implement an autonomous system capable of following a reference, even when the dynamic changes. We also implement adaptive control to this system, which enables the update of the models and controllers when the currently observed behavior does not match any of the known models.

Keywords

Nonlinear system; Multiple model; Multiple controllers; Supervisor; System identification; Offset estimation.

Resumo

Nesta dissertação, apresentamos um exemplo de uma aplicação para controlar um sistema não-linear usando o método de modelos múltiplos. O sistema tem uma dinâmica variável, onde cada dinâmica pode ser representada por um modelo. Para cada modelo é projectado um controlador. Como a dinâmica do sistema é manipulada externamente, e portanto não pode ser medida directamente, é usado um supervisor para detectar as mudanças na dinâmica.

Para controlar um sistema não-linear, primeiro devemos identificar algumas características que nos permitiram criar um modelo. Como um sistema não-linear tem várias dinâmicas incrementais dependentes do ponto de funcionamento, é identificado um conjunto de modelos, onde cada modelo corresponde a um ponto de funcionamento específico. Com os modelos múltiplos obtidos a partir da identificação do sistema, podemos projectar um controlador formado pelo agrupamento de controladores locais, em que cada um corresponde a um modelo local. O supervisor detecta qual o modelo mais semelhante ao comportamento actual do sistema e escolhe o controlador correspondente. Os *offsets* do modelo também são estimados durante as experiências. Um algoritmo original que aborda a presença de *offsets* desconhecidos é proposto e demostrado experimentalmente.

Depois de projectar os controladores e o supervisor, implementamos um sistema autónomo capaz de seguir uma referência, mesmo quando a dinâmica altera. Também aplicamos controlo adaptativo a este sistema, o que permite a actualização dos modelos e dos controladores quando o comportamento actual do sistema não corresponde a nenhum dos modelos conhecidos.

Palavras Chave

Sistema não-linear; Modelos múltiplos; Controladores múltiplos; Supervisor; Identificação de sistemas; Estimação de *offsets*.

Contents

List of Figures

List of Tables

List of Abbreviations

ARMAX	Auto Regressive Moving Average with eXogenous input
ARX	Auto Regressive with eXogenous input
KF	Kalman Filter
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
RLS	Recursive Least Squares

Symbol	Meaning	Unit
δu	Command action from LQG controllers	$[V]$
Δ	Difference operator (discrete domain)	$-$
θ	Throttle opening	[°(degree)]
ρ	Density	$[kg.m^{-3}]$
σ	Supervisor output	$-$
σ_o	Offset supervisor index	$-$
σ_t	Transient supervisor index	$-$
τ	Delay from heater to sensor	$[s]$
τ_D	Controller dwell-time	$[s]$
A	Section of the tube	$[m^2]$
c_p	Specific heat capacity	$[J.kg^{-1}.K^{-1}]$
C	Heat capacity	$[J.m^{-3}.K^{-1}]$
K_u	Gain of heater	$[A]$
l	Distance from heater to sensor	$[m]$
$\mathcal{M}$	Set of models	$-$
P	System input power	$[W]$
q	Flow of air in the tube	$[m^3.s^{-1}]$
T	Temperature after heater	[℃]
T_a	Atmosphere temperature	[℃]
u	Heat regulator (input)	$[V]$
v	Velocity of air in the tube	$[m.s^{-1}]$
V	Volume	$[m^3]$
y	Measured temperature (output)	$[V]$

Chapter 1

Introduction

1.1 Motivation

Control systems are becoming more and more important nowadays, where the demand for systems with more accurate, robust and faster performance is growing more exigent day by day. Primary systems that previously assisted in simple tasks now have become rather advanced, capable and essential tools, indispensable in every area from industry to medicine and economy. Moreover, we can project these systems to be autonomous, thereby implying a diversity of ways to improve system performance.

A system can have one or more actuators and sensors. The actuators and sensors are associated to the inputs and outputs of the system. Corresponding data enable the system to be modeled and, afterwards, its performance can to be predicted and controlled. As the knowledge about control systems is expanding, a higher number of solutions are presented. When studying control systems, we need to overcome many obstacles like uncertainty, nonlinearity, time variation and complexity, which lead to several approaches and design methods. Still, in order to have a reliable and quality outcome, there is a trade-off when choosing how many approaches we use and the complexity of the solution. If we choose a simple approach, the performance of the system model can be underestimated and, thereby, unable to simulate the essential dynamic of the system. If the approach is too complex, in addition to having an overestimated performance leading to an undesirable outcome, the manipulation of a complex system itself is difficult for both designer and user.

Almost every system can be considered as nonlinear, even if they seem really simple. Most of the times, there is a nonlinear behavior in these systems, whether in systems where the dynamic can change abruptly or in simple systems that seem close to being linear. Examples as a solar power plant [1], patient anesthesia [2] and flight control [3], among others, all have a part of its dynamic as nonlinear that must not be ignored and all need to be controlled.

In this project, we study the nonlinearity of a system and how we can control it, based on some reliable approaches. To identify a nonlinear system, a possible approach to use is local model identification for further construction of the global model. In this class of methods, different local models are identified to represent the system dynamics around representative operating points and then interpolate the local models when the system works around other points. Then, we design a local controller for each local model and a supervisor to identify the current model and activate the corresponding local controller. We perform some real experiments on how the number of models influences the controlled system performance and we also do some simulations with adaptive control.

1.2 Formulation of the Problem

To identify a nonlinear system, a possible approach to use is local model identification for further construction of the global model. In this class of methods, different local models are identified to represent the system dynamics around representative operating points that are then interpolated when the system works around other points.

The system to be identified is the Process Trainer PT 326 from Feedback Instruments Ltd., linked to the computer through a PCI-MIO-16E-4 data acquisition board, consisting in a laboratory scale air heating system. In the system in study, the air passes through a tube where it is heated and its temperature measured. The airflow can be changed through the manipulation of a throttle opening.

Nonparametric and parametric identification are used to compute the system model, after the analysis of some system characteristics. As we are dealing with a nonlinear system, instead of one, we may consider its approximation by multiple models. Each model, named as local model, corresponds to a specific operating point, around which the nonlinear system has a specific dynamic.

With the models obtained, we are able to design a controller in order to make that system perform as desired. The question to design a controller able to control the nonlinear system is then raised. In this work, we try to answer that question by using multiple controllers. Each controller is designed for a unique local model and a switch circuit will choose which controller to use.

Because the system dynamics depends on an unknown variable, the flow that can be manipulated but cannot be directly measured, a supervisor must be used to estimate the current model. Therefore, we intend to have a controller switch manipulated by the supervisor, in order to increase the performance.

The methodology used to design local controllers is the *Linear Quadratic Gaussian* (LQG) controller. A *Local Model Network* (LMN) is also designed to help to adjust the LQG controllers and supervisor parameters.

1.3 State of the Art

With the objective of studying, stabilizing and predicting the behavior of a large variety of systems, researchers try to create new ways and methods to overcome the problems encountered. Some introductory bibliography of these studies and researches can be found in [4, 5, 6, 7].

Unlike linear systems, nonlinear systems are more difficult to deal with. Sometimes, we can choose a region of the nonlinear system that works as a linear system. In other cases, this way may be ineffective and we must deal with the systems nonlinearity. In [8, 9, 10, 4, 11], some approaches are presented to deal with nonlinear systems.

As mentioned before, one of the approaches for nonlinear systems is the use of multiple local models [12, 11, 2, 13]. Various approaches to identify systems take place in [4, 6, 7], where we can find the parametric and non-parametric identifications. Some studies on the Process Trainer PT 326 can be seen in [14, 15]. Still, no studies concerning the use of multiple dynamics or different flows of this specific system were found.

There is a wide research on the design of controllers for modeled systems, which can be found in [6, 7]. Some bibliography on the controller we use, the LQG, is also in [16, 17, 18].

The use of adaptive control to achieve better performances on the controlled systems has gained interest lately [19, 20]. The application of multiple models with adaptive models can be a useful approach with vast applications in control systems, as in [1, 21].

1.4 Main Contributions

Based on the importance of the utility of nonlinear systems on a vast number of situations, the interest on improving the known approaches, like multiple models, increase. In this thesis, we have as an objective to control a real nonlinear system using the multiple model approach.

First, we present some characteristics of the system such as noise, hysteresis and nonlinear behavior and then we present how to obtain each local model. We present a method to group the models into a global model, in order to simulate the system behavior. Moreover, we compute multiple controllers to compensate the multiple models and we design a supervisor to detect the current model.

The supervisor is based on two supervisors, where one compares the system transients with the estimated models transients, named transient supervisor. The other supervisor, named as offset supervisor, calculates the distance, or offset, from the current system to the estimated models. Moreover, an original method to estimate the offset for the estimated models is presented.

In addition, we demonstrate an example where we apply adaptive control, with the performance achieved using the method mentioned before. The performance is then tested when some models are missing.

1.5 Dissertation Outline

The dissertation is structured in the following chapters. After the introduction chapter, where we introduce the problem and some contributions of this work, we have chapter 2 where we present the equipment and its characteristics.

First, we present the theoretical mathematical model, as well as some solutions for starting issues relating the sensor measurements. Then, we use nonparametric and parametric identification to compute local models and, in the last section, we present the offset estimation algorithm.

In the chapter 3 we propose to design local controllers with the *Linear Quadratic Gaussian* (LQG) method and the supervisor to be a combination of two supervisors, each comparing different characteristics of the system dynamic. After the design of the controllers, experiments are done to compare the performance of the system with multiple controllers to a single one. Moreover, we do some experiments to compare the performance of the system when using the supervisors individually and together.

In chapter 4 we present a brief implementation of *Adaptive Control*, where we use the *Recursive Least Squares* (RLS) algorithm to estimate the current model parameters. Simulations on the system performance are done and analyzed when using less estimated models. The conclusion and future work suggestions are given in Chapter 5.

Chapter 2

Identification

2.1 Equipment

The plant considered in this project is an air heating system (figure 2.1). The air atmosphere is drawn by a blower (on the left), passes through a heater (①) and a tube (②) before returning to the atmosphere. The process consists in heating the air at a starting point u and measure its temperature T at a terminal point y with a thermocouple sensor.

The airflow q can be manually manipulated by changing the throttle opening (θ°) from 10° to 165° (degrees), which has an effect on system dynamic behavior. A minimum throttle opening ($\theta_{min} = 10$°) corresponds to minimum flow and a maximum throttle opening ($\theta_{max} = 165$°) to maximum flow.

2.1.1 Equipment Description

Specifications from the PT 326 manual [15] informs that both input and output voltage have ranges from $0\,V$ to $10\,V$ (volts). The scale for the output temperature goes from 0℃ to 80℃, but the controlled air temperature range is only from 30℃ to 60℃. Furthermore, the air velocity v is indicated as being from $1\,ft.s^{-1}$ to $10\,ft.s^{-1}$, equivalent to values of $v_{min} = 0.3048\,m.s^{-1}$ to $v_{max} = 3.048\,m.s^{-1}$ in SI units. Finally, the range of the heat power goes from $15\,W$ to $80\,W$ (which will be important for further analysis).

In figure 2.1 we indicate the distance from the heater to the sensor as $l = 0.279\,m$ and the diameter of the tube as $d = 0.046\,m$. The tube has the form of a cylinder, where the section can be calculated with $A = \frac{\pi d^2}{4} = 16.6 \times 10^{-4}\,m^2$. Therefore, we are able to calculate the airflow $q = Av$, which means its range goes from $q_{min} = 0.507 \times 10^{-3}\,m^3.s^{-1}$ to $q_{max} = 5.07 \times 10^{-3}\,m^3.s^{-1}$.

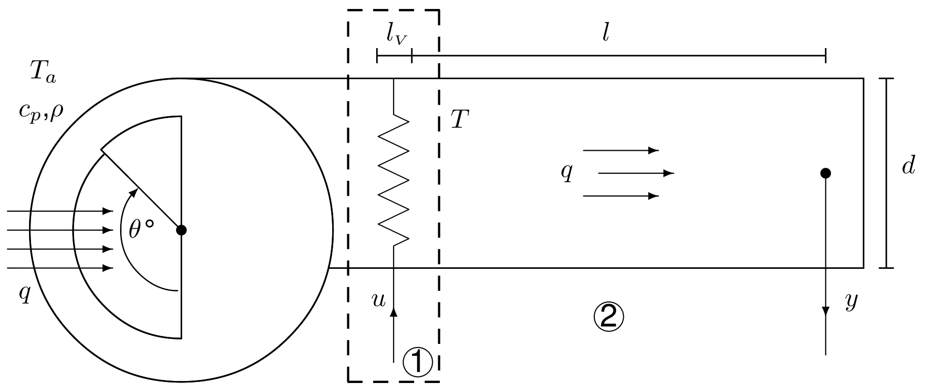


Figure 2.1: Scheme of air heating system used on the experiment.

We have also the air characteristics in the atmosphere, like the room temperature T_a, the air density ρ and the air specific heat capacity c that was not measured.

2.1.2 Process Physical Model

The heat process in analysis can be divided in two parts, as figure 2.1 shows. Part ① is similar to the oil heating in a solar collector field, where the exchange of energy can be described by

$$CA\Delta x\big[T(x,t+\Delta t)-T(x,t)\big] = Cq\Delta t\big[T(x,t)-T(x+\Delta x,t)\big]+P(t)\Delta t\,, \quad (2.1)$$

where $C = c_p\rho$ is the heat capacity of the air, the temperature at the heater exit is T, the flow of air inside the tube is represented by q and P is the power of the heat injected in the system. The space and time variables are represented as x (in meters) and t (in seconds). Appendix A shows how to get to equation (2.1).

To analyze the process for different flows, we use a low order transfer function. We assume $\Delta x = l_V$ as a constant and then we can express the air volume heated as $V = Al_V$, also we have the input power $P(t) = K_u u(t)$ where $u(t)$ is the input voltage and K_u is the gain.

Having Δx constant, the temperature $T(x,t)$ no longer depends on x and can be written as $T(t)$. Dividing equation (2.1) by Δt and making $\Delta t \to 0$ we get the following equation in $[W]$

$$CV\,\frac{\partial}{\partial t}T(t) = Cq\big[T_a(t)-T(t)\big]+K_u u(t) \quad . \quad (2.2)$$

Assuming negligible the heat losses and that $\tau = \frac{Al}{q}$ is the delay from the heater to the sensor, in part ② the temperature at y is the same as T delayed by τ seconds, or rather

$$y(t) = T(t-\tau) \quad . \tag{2.3}$$

Applying the Laplace transform to equations (2.2) and (2.3) and dividing the first by CV we can get to

$$\begin{cases} sT(s) = \dfrac{q}{V}\big[T_a(s) - T(s)\big] + \dfrac{K_u}{CV}U(s) \\ Y(s) = e^{-s\tau}\,T(s) \end{cases} \quad . \tag{2.4}$$

Putting together the two equations in (2.4) and assuming $T_a(s) = 0$ we have

$$Y(s) = \frac{\frac{K_u}{Cq}e^{-s\frac{Al}{q}}}{\frac{V}{q}s+1}U(s) = \frac{Ke^{-s\tau}}{T_p s+1}U(s) \quad . \tag{2.5}$$

This is the transfer function of a heating and ventilation system. The dynamic process is affected by the airflow q in the way that, when it increases, the gain K, time constant T_p and delay τ decreases. Opposite to the time constant T_p, we have the bandwidth LB which increases with the airflow.

In our situation, T_a is non-zero, but we can assume it is constant during the process, which generate an offset on the output, but without influencing the incremental dynamic of the system.

2.1.3 Experimental Preparation

All experiments were performed at room temperature between 25℃ and 30℃, since greater variations can influence directly the measured values, thereby changing the system behavior and a model misidentification.

Each experiment consists in the generation of an input signal to be sent to the heater and the data acquisition of the temperature measured by the sensor. In the beginning, the system must be turned on and kept at constant flow ($\theta = 40°$) and an input of $5\,V$ for 30 minutes, as a warm up, to drive the system temperature to a required condition.

There is also an overheating problem if the system works with high temperatures for too long and data acquired in these conditions are undesirable. To avoid the waiting time for the system to cool down, we start by having most experiments happening in low temperatures.

Moreover, we encountered an irregular perturbation in the signal that is shown in figure 2.2, where we can see six look-like peaks without a regular period. These peaks can greatly corrupt the noise characteristics, such as mean, variance and autocorrelation. Since it seems a nonlinear perturbation, it can extend to some errors in identification analysis, especially in parametric

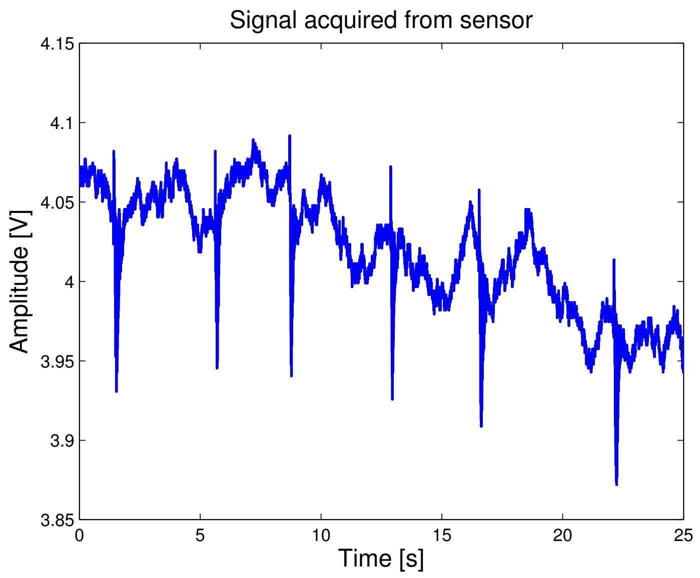


Figure 2.2: Stationary signal for maximum flow and input at $0\,V$.

identification, whereas it is expected linear noise.

An issue coming from this problem could be when using a lower sample frequency, where bad samples belonging to the peaks could be collected, influencing subsequent analysis. Figure 2.3 shows how this problem may be overcome with a median filter.

To prevent this situation we need to filter the signal for any further data acquisition. Section 2.1.4 have more information about the filter used.

2.1.4 Filter Design

As saw in section 2.1.3, we are facing nonlinear perturbations, which must be filtered to avoid errors in the upcoming identification. We can guess a nonlinear filter could be better than a linear filter, but we try both to figure out which one is better in terms of outlier rejection as well as less delay induced. The expected signal (the one to be compared), is a signal without peaks, computed by a median filter.

We try a linear low-pass filter in the form $H_c(s) = \frac{1}{T_c s+1}$ with the cutoff frequency $\omega_c = \frac{1}{T_c}$. Our objective is to find the cutoff frequency that better fits the expected signal, with the lowest possible delay. Therefore, we have a system with the real output signal as the input, $Y_c(s)$ and the output is the signal expected, $Y(s)$, and we have the input-output relation $Y(s) = H_c(s)Y_c(s)$.

For the nonlinear filter, we use a moving median filter with windows of size

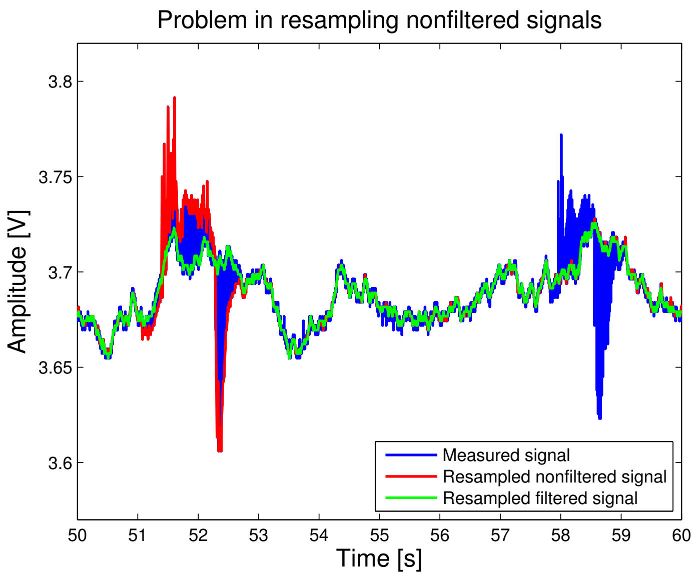


Figure 2.3: The real signal (blue), signal resampled with sample frequency of $100\,Hz$ (red), and signal filtered by moving median with order 5 resampled with sample frequency of $100\,Hz$ (green), for maximum flow.

3, 5 and 7, each one corresponding to a delay of 1, 2 and 3 samples. Then, we compute the error between the expected signal and each filtered signal to obtain figure 2.4, where we can compare this errors. Based on the result, we can conclude that the moving median filter with a window of 5 samples is the best choice, and will be used from now on.

We can also look to the Fourier transform in figure 2.5 where we can identify that every multiple of $100\,Hz$ highlights in the entire spectrum for the nonfiltered signal and disappears for the filtered signal. Also $50\,Hz$ and $150\,Hz$ appears that could correspond to electrical grid frequency.

The $50\,Hz$ noise does not have much effect on the signal as the perturbation mentioned before, and to avoid more delay, we stop here. If necessary, for upcoming analysis, the signal can be filtered with a low-pass filter with a convenient cutoff frequency depending on the bandwidth of the system.

2.1.5 Sensor Characteristics

Before proceeding to identification we need to analyze some characteristics about the temperature sensor, such as noise and the relation between the electric voltage and the temperature measured.

Using the temperature scale from the equipment and evaluating the output voltage, we can obtain the sensor voltage-temperature relation. The result, as

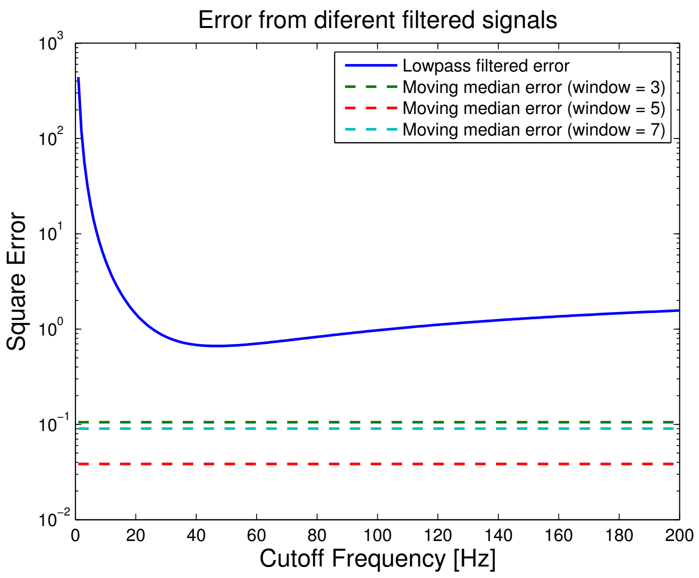


Figure 2.4: Error of low-pass filtered and moving median filtered signals for throttle opening of $40°$.

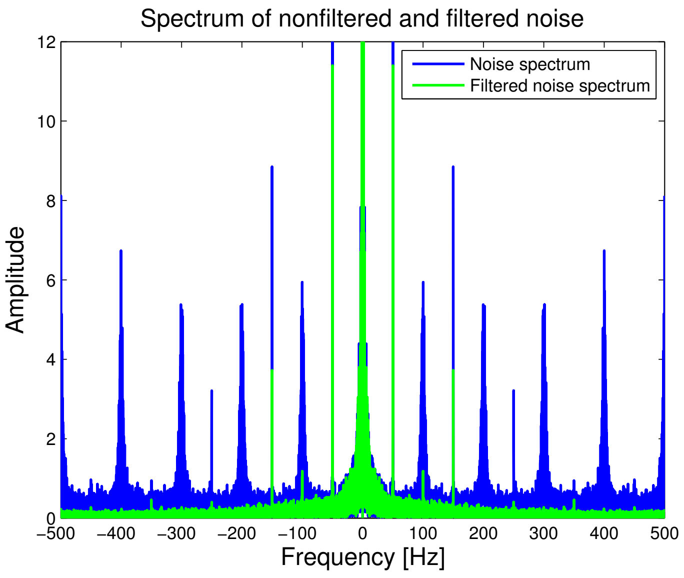


Figure 2.5: Spectrum of unfiltered and filtered signal using the Fourier transform for maximum flow.

displayed in figure 2.6, shows that this relation is nonlinear (blue[1]). However, we can have a linear approximation (green) from 30℃ to 50℃, that is within

[1] The blue signal is computed with the 'pchip.m' function in *Matlab*.

the controlled temperature range denoted by the manufacturer, which means that only experiments with output values above $3\,V$ can be used.

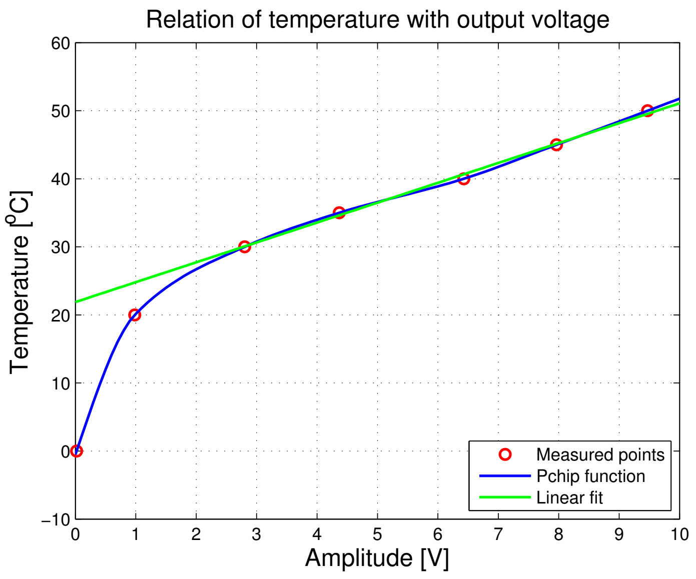


Figure 2.6: Relation between voltage and temperature at the output.

To know how the output signal is affected by noise, we analyze the stationary signal in time domain and the autocorrelation, for minimum and maximum throttle opening. Figure 2.7 shows this two signals, where we can notice that the left scale (blue) is higher than the right scale (green), having the signal an higher variance for minimum flow.

Performing the histogram and autocorrelation for the two signals, we get figures 2.8 and 2.9. Analyzing the histogram and autocorrelation for both signals, we can state that we are dealing with correlated noise, thus excluding the presence of white noise. In addition, the standard deviation is $124.7\,mV$ and $16.5\,mV$ for minimum and maximum flows respectively.

2.2 System Identification

To be able to control a system, we need a model that represents its dynamic. We model the global nonlinear system by modeling the dynamic of each local model, corresponding to a certain regime of values for the throttle opening, and then by patching them together.

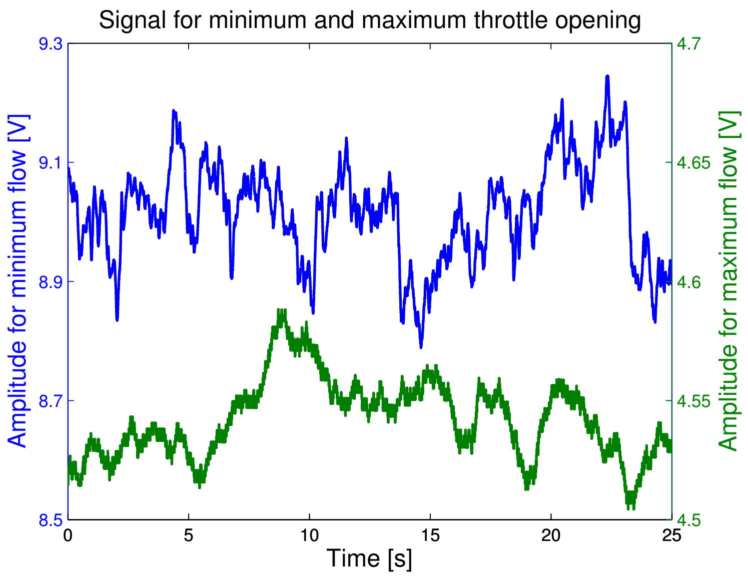


Figure 2.7: Noise signal for minimum (blue) and maximum (green) flows.

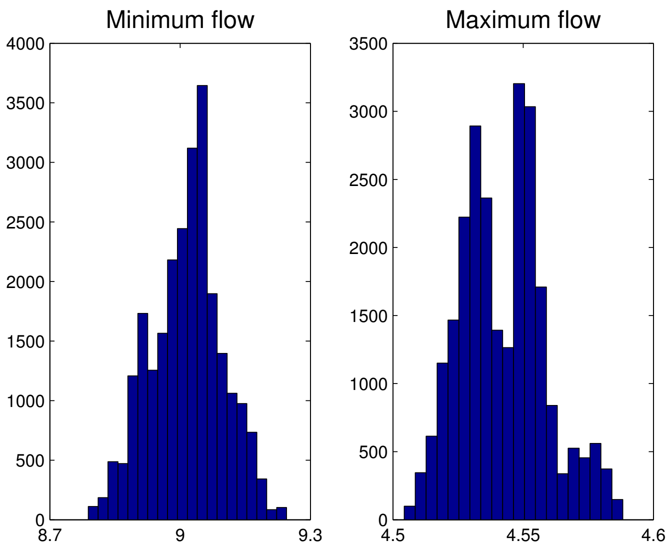


Figure 2.8: Histograms of noise to minimum and maximum flows.

2.2.1 Modelling the Global System

To know which are the values of throttle openings that we need to choose to proceed to identification analysis, we first need to know how the throttle opening affects the flow, or more precisely, how it affects the process dynamic. For that we input a ramp signal with positive and negative slope, to know the

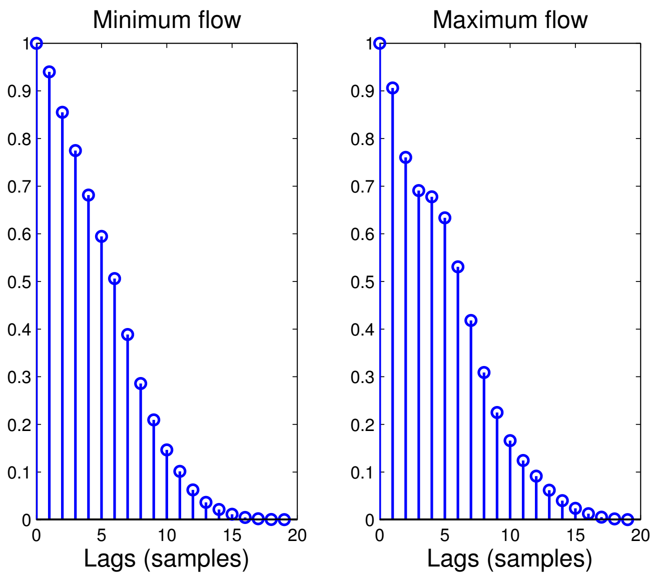


Figure 2.9: Autocorrelation of noise to minimum and maximum flows for the first 20 samples.

temperature hysteresis effect, that is shown in figure 2.10.

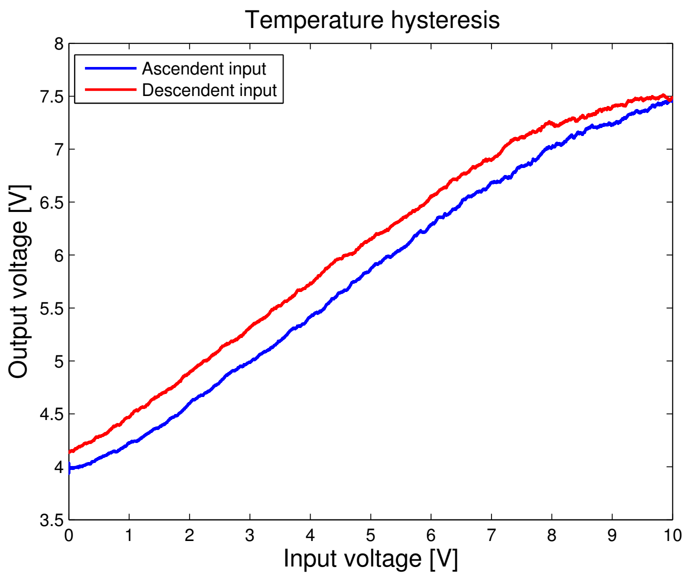


Figure 2.10: Temperature hysteresis for a throttle opening of $130°$.

Here we can see the differences in the heating and cooling processes that can be explained by the different conditions that each one goes. The cooling process depends on the heat transferred from the heater and from the heated equipment to the air in the tube, while the heating process only depends on the first (and that is why the output is lower for the same input voltage). However, this effect seems too small and to simplify, is not taken into account.

Then, we injected at the input a ramp signal to know the static characteristic for throttle openings starting at $10°$, every $10°$, to $150°$ and including $165°$, as shown in figure 2.11. We can observe that, for small flows, the output increases to higher values until it saturates (due to equipment restriction), and for large flows, the dynamic have slight changes. Based on the static characteristic we chose the input range between $0.5\,V$ and $2.5\,V$, since a higher input leads to output saturation and around $0\,V$ there is also a saturation due to the minimum temperature (and is less controllable).

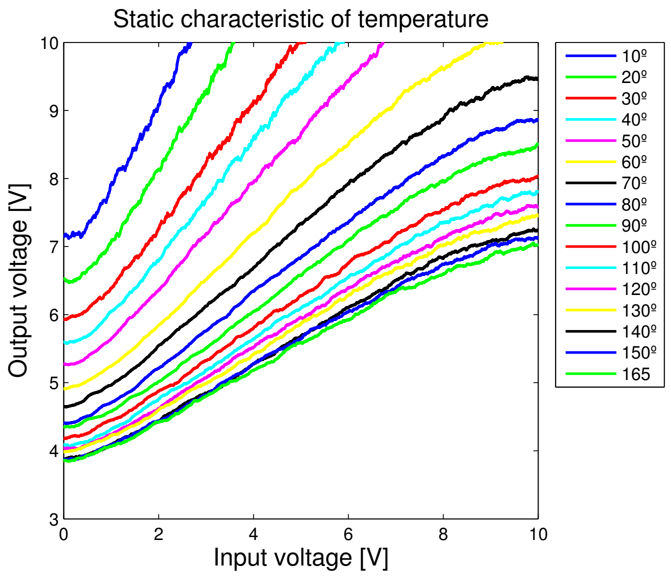


Figure 2.11: Static characteristic of temperature for sixteen throttle openings (the order of the legend corresponds to the order of signals at $0\,V$).

Also on figure 2.11, we can notice that, for the input of $0\,V$, the different outputs are decreasing with the increasing of the throttle opening (or flow). This happens because the minimum heat power input is $15\,W$, which means the air is being heated even for input at $0\,V$, and the output values follows the relation of gain K in equation (2.5).

After choosing the input range, we generate a squared wave signal, between $0.5\,V$ and $2.5\,V$, to inject in the system and estimate the model, to match equation (2.5), using the *Prediction-Error Minimization* ('pem.m') method from *Matlab System Identification tool (Ident)* [4]. Figure 2.12 represents the computed values for each throttle opening, and based on the results, we can choose how many models will cover the global system, including which throttle openings will be used to identification.

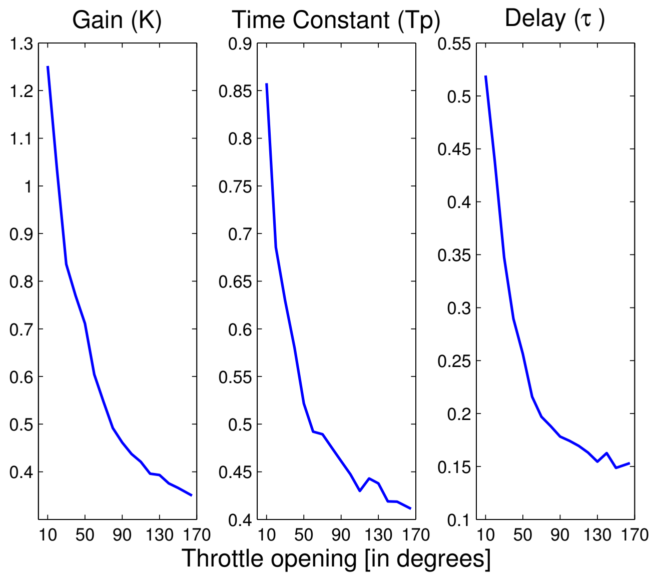


Figure 2.12: System parameters for each throttle opening, according with equation (2.5).

Analyzing the three graphics in figure 2.12, we can identify that every parameter values decreases when the flow increases, as expected from equation (2.5). We can also verify that for higher flows, the parameters differ less than for smaller flows, which means that small throttle openings have greater differences from their neighbor than higher throttle openings. Based on this and for identification purposes, we choose three throttle openings in a way that better covers the global system that are $30°$, $70°$ and $130°$.

To perform identification, nonparametric and parametric methods implemented in *Ident* were used. That allow us to get the bandwidth and the local model for each throttle opening (θ°) of the process, each to be represented by a discrete transfer function, as figure 2.13 shows.

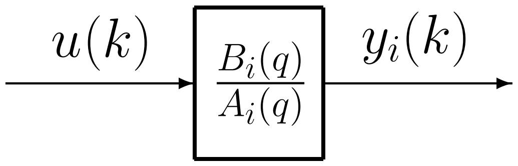


Figure 2.13: Model of system to be identified.

2.2.2 Nonparametric Identification

This section aims at identifying the main aspects of the system such as bandwidth, gain and delay to find the appropriate sample frequency for the parametric identification. For this purpose we used spectral analysis and correlation analysis in order to obtain the frequency and time responses [5].

Since the system is stable in open loop, the design and identification of a closed loop system is discarded for simplicity. The purpose is then to achieve a discrete model with the behavior equivalent to the real system.

A sample frequency of $1000\,Hz$ is applied to the input of the heating system and the responses of the system temperature are observed, for the three flows considered. Figure 2.14 shows the response of the three chosen throttle openings for the same square wave input signal. We can observe that, for the same input, each output signal has a different offset, which is known as the operating point.

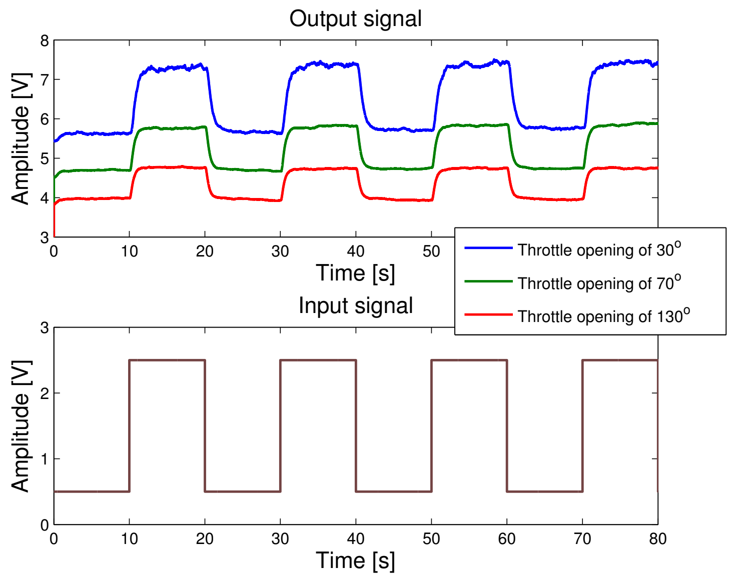


Figure 2.14: System response to square wave input signal for the three throttle openings and sample period of $1\,ms$.

For further analysis, the average of each signal is subtracted, to eliminate the offset. The time response (correlation analysis) is calculated from a set of samples containing a step example within the square signal, as illustrated in the figure 2.15. The frequency response (spectral analysis) is calculated for the square input signal with the method *Spectral Analysis Estimates with Frequency Dependent Resolution* for 6000 frequencies (figure 2.16).

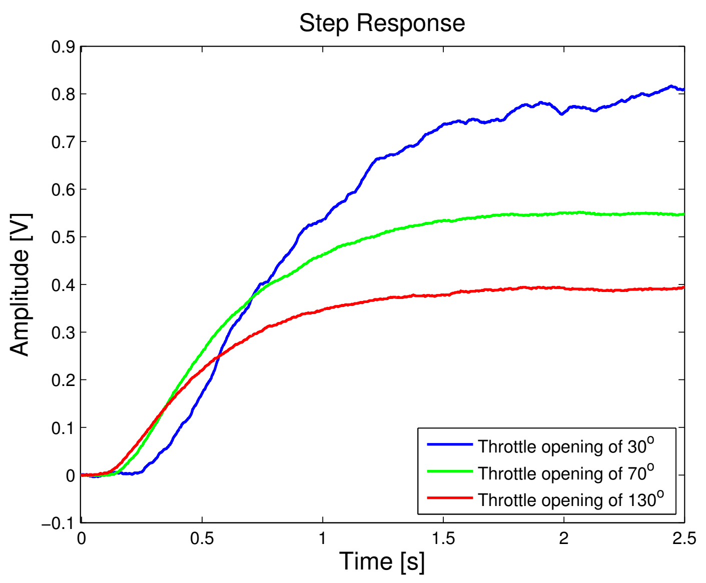


Figure 2.15: System step response obtained by correlation analysis.

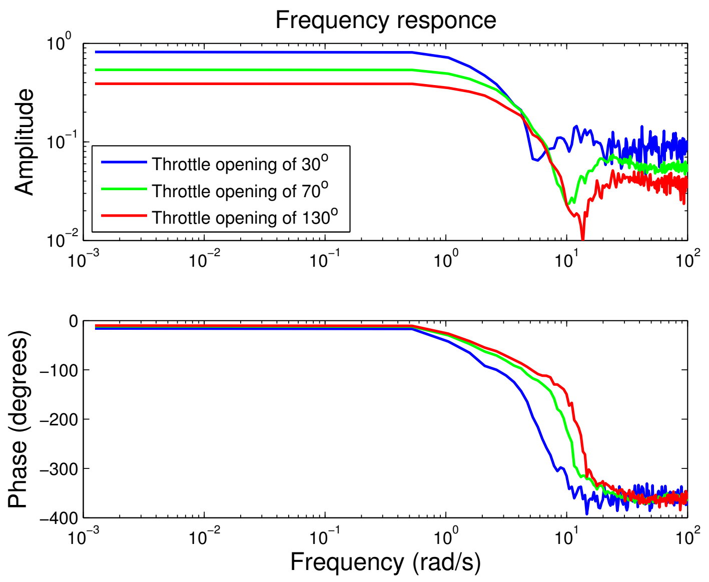


Figure 2.16: System frequency response obtained by spectral analysis.

From the observation of figures 2.15 and 2.16, we can check once again the differences and the relationship between the results and the variation of airflow, *i.e.*, when increasing airflow, bandwidth increases and the time constant, gain and delay decrease. This influence of the airflow is the reason why the global system cannot be considered linear.

Table 2.1 presents the values of the delay, gain and bandwidth, plus the sampling frequency which is explained below.

The delay is calculated as the time interval from the time $0\,s$ (time of initiation of the step) until the start of the system response. Figure 2.17 is a close view of the initial part of the step response unit (figure 2.15) to assist the measurement of the delay.

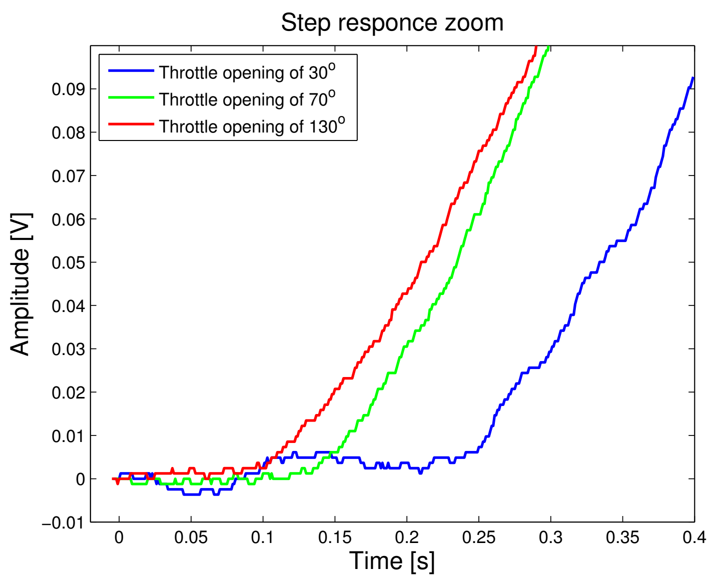


Figure 2.17: System step response zoom from figure 2.15.

The gain is calculated by dividing the amplitude of the output signal, of the square wave response, by the input signal amplitude (figure 2.14).

The frequency where the response decays $3\,dB$ (figure 2.16) is the bandwidth and from it we can estimate the appropriate sample frequency (f_s) for the system parametric identification. First we calculate the frequency of the bandwidth $f_{LB} = \frac{LB(rad.s^{-1})}{2\pi}\,Hz$. Then a practical rule used is that the sampling frequency should be about 10 to 30 times the bandwidth ($-3\,dB$) of the system under analysis [6], which means that

$$10\,f_{LB} \leq f_s \leq 30\,f_{LB} \quad . \tag{2.6}$$

All the values mentioned before are shown in table 2.1[2]. The last column presents a range of sampling frequencies appropriate for each throttle opening degree, to later proceed to the parametric identification.

[2] The symbol $\sim$ means the value is approximated and could not be precisely measured.

Table 2.1: Values of delay, gain, bandwidth and corresponding frequency and in the last column the range to the sampling frequency.

$\theta°$	Delay$[ms]$	Gain	$LB[rad.s^{-1}]$	$f_{LB}[Hz]$	$10f_{LB} \leq f_s \leq 30f_{LB}$
30	~ 0.25	0.84	1.581	0.252	$2.52 \leq f_s \leq 7.55$
70	~ 0.15	0.57	2.057	0.327	$3.27 \leq f_s \leq 9.82$
130	~ 0.10	0.40	2.369	0.377	$3.77 \leq f_s \leq 11.31$

The range for choosing a sampling frequency suitable to operate in the global system is $3.77\,Hz \leq f_s \leq 7.55\,Hz$ corresponding to a sample period of $0.13\,s \leq h_s \leq 0.27\,s$. However, we try different sampling periods h_s to verify what is really the best one.

Being the maximum bandwidth $LB_{max} = 2.37\,rad.s^{-1}$, we can put, after the moving median filter, a low-pass filter with the cutoff frequency a decade above the bandwidth, *i.e.*, $f_{LP} = 30\,rad.s^{-1} > 23.7\,rad.s^{-1} = 10\,LB_{max}$.

We can measure the flows that we are dealing with for more information, knowing $q = \frac{Al}{\tau}$ and using A and l mentioned in section 2.1 and the delay obtained, we have $q_{30^o} \approx 1.85 \times 10^{-3}\,m^3.s^{-1}$, $q_{70^o} \approx 3.09 \times 10^{-3}\,m^3.s^{-1}$ and $q_{130^o} \approx 4.63 \times 10^{-3}\,m^3.s^{-1}$, which values are within the flow range presented also in section 2.1.

2.2.3 Sample Frequency Analysis

Knowing the bandwidth for each throttle opening, we can apply a PRBS signal with a sample period of $1\,ms$ to the input, where the band B of the signal is lower than the lowest time constant, $B = 0.4 < {LB_{max}}^{-1} = 2.37^{-1} = 0.42$. With both input and output, we can estimate the *ARMAX* model, with orders $[n_a\, n_b\, n_c\, n_k]$ such that

$$\begin{aligned} y(k) + a_1\, y(k-1) + \ldots + a_{n_a}\, y(k-n_a) = \\ = b_1\, u(k-n_k) + b_2\, u(k-n_k-1) + \ldots + b_{n_b}\, u(k-n_k-n_b+1) + \\ + e(k) + c_1\, e(k-1) + \ldots + c_{n_c}\, e(k-n_c)\ , \end{aligned} \quad (2.7)$$

where $y(\xi)$ and $u(\xi)$ are the output and input samples obtained for the system performance, $e(\xi)$ is white Gaussian noise and a_ξ, b_ξ and c_ξ are the model parameters to estimate.

We resampled the $1\,ms$ sample time PRBS signal to different sample rates $[h_1, h_2, ..., h_s, ..., h_n]$ and estimated the parameters of the *ARMAX* model with orders $[2\,1\,2\,n_{k_s}]$ for each one. The delay parameter needed to be calculated for each sample time, as $n_{k_s} = fix\left(\frac{delay_\theta}{h_s}\right) + 1$, *i.e.*, the delay order must be such that the result delay ($h_s \times n_{k_s}$) are the closest possible, for different sampling periods. An example is, with $delay_{70^o} = 0.15\,s$, $h_1 = 0.1\,s$ and $h_2 = 0.05\,s$, the delay n_k would be $n_{k_1} = 2$ and $n_{k_2} = 3$, which means a result delay of $0.2\,s$ and $0.15\,s$ respectively.

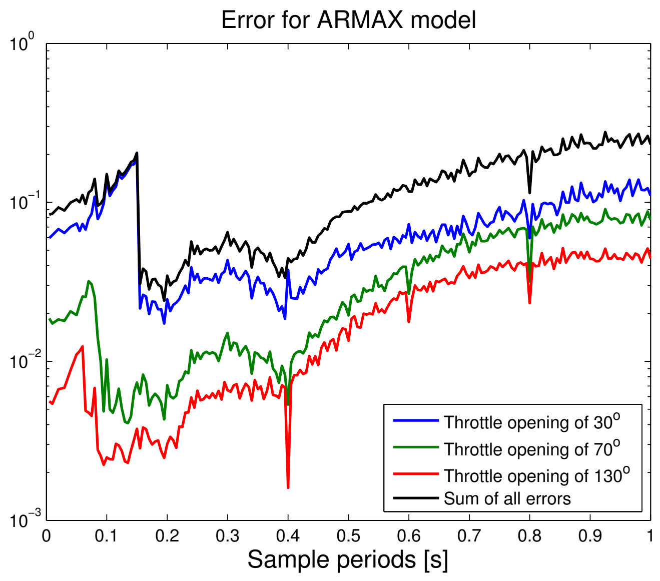


Figure 2.18: Model errors for various sampling periods for each throttle opening used and sum of errors for each sampling period.

Figure 2.18 presents the results of the errors for each throttle opening (blue, green and red), and the sum of errors of each sample period (black), where we can conclude the less error is attain for sampling periods between about $0.15\,s$ and $0.5\,s$. This result can be explained for bigger sampling periods, because even the lower delay achievable can be much higher than the real delay, influencing the parameters estimation, and for smaller sampling periods, because the orders used $[2\,1\,2\,n_k]$ may be too low for the model to recognize and estimate the incremental dynamic of the system.

Since this method was not performed to every *ARMAX* orders nor for every sampling periods, alone it is not enough to choose a sample time. Nevertheless, since this result and the sampling period range obtained in section 2.2.2 are consistent, we can choose thereby the sampling period as $0.2\,s$, which means a sampling frequency of $5\,Hz$.

2.2.4 Parametric Identification

For parametric identification of the system the *ARMAX* structure of equation (4.2) is also chosen, since it leads to a better model with correlated noise than ARX. To estimate the best model, a $400\,s$ duration PRBS signal is chosen, where we use the first $300\,s$ for parameter estimation and the last $100\,s$ for validation. With this we intend to approximate the response of the estimated model to the real system response. The sample time used is $0.2\,s$ for the input and output signal, while the command signal to the process is kept at $1\,ms$.

Several orders of the *ARMAX* model $[n_a\,n_b\,n_c\,n_k]$ were used, limiting $n_a = n_c$, and tested several values for n_a and n_b between 1 and 3 (above 3 it can overestimate the noise) and n_k between 0 and 3 (where 3 corresponds to a delay of $0.6\,s$).

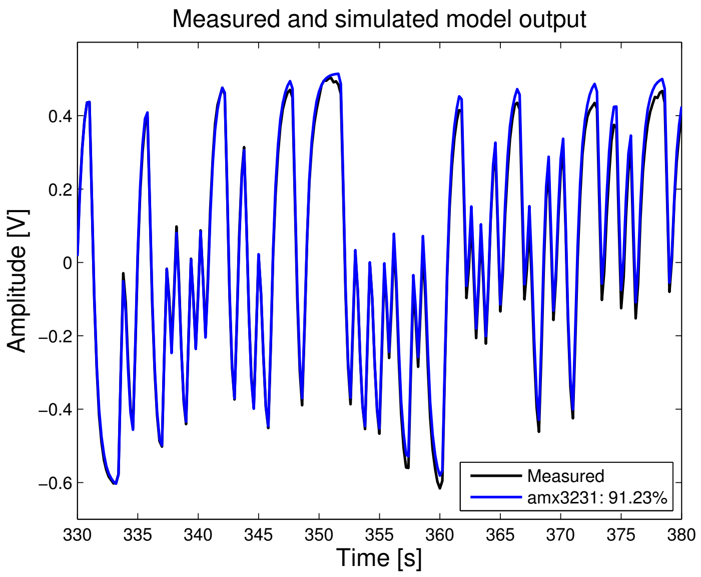


Figure 2.19: Model and simulated output to PRBS validation signal, with the best fit for a throttle opening of $70\,°$.

Table 2.2 presents the model parameters and order corresponding to the best fitting percentage for each throttle opening degree. The offset is the operating point around which each model is computed. When analyzing the various models, some models had best fits than the ones in table 2.2, but they had a low performance in step response and were discarded. By observation of figure 2.19, the response of the 'amx3231' model appears to be a reasonable approximation of reality.

From table 2.2, we can verify that the order of poles and zeros is respectively 3 and 2, for every throttle opening and only the delay changes.

For the best analysis of each throttle opening, figure 2.20 shows the simulated response to the unit step. We can see that the first $2.5\,s$ are very similar to the signals in figure 2.15, and the gain and delay values of the estimated models are also close to the ones in table 2.1.

Table 2.2: Parameters and percentage of best fits of the *ARMAX* model, to each throttle opening used, including the offset.

θ	Best fit (%)	Order $[n_a\, n_b\, n_c\, n_k]$	Offset $b\,[V]$	*ARMAX* Model $A(q)y(k) = B(q)u(k-n_k) + C(q)e(k)$
$30°$	88.43	$[3,2,3,2]$	6.54	$A(q) = q^3 - 1.1310q^2 + 0.3862q^1 - 0.0596$ $B(q) = 0.0562q^1 + 0.1103$ $C(q) = q^3 - 0.2480q^2 + 0.2360q^1 + 0.0983$
$70°$	91.23	$[3,2,3,1]$	5.25	$A(q) = q^3 - 1.0264q^2 + 0.3446q^1 - 0.0576$ $B(q) = 0.0148q^2 + 0.1268q^1$ $C(q) = q^3 + 0.4755q^2 + 0.3112q^1 + 0.2060$
$130°$	92.96	$[3,2,3,1]$	4.32	$A(q) = q^3 - 0.8513q^2 + 0.1953q^1 - 0.0247$ $B(q) = 0.0276q^2 + 0.0981q^1$ $C(q) = q^3 + 0.5694q^2 + 0.4129q^1 + 0.2838$

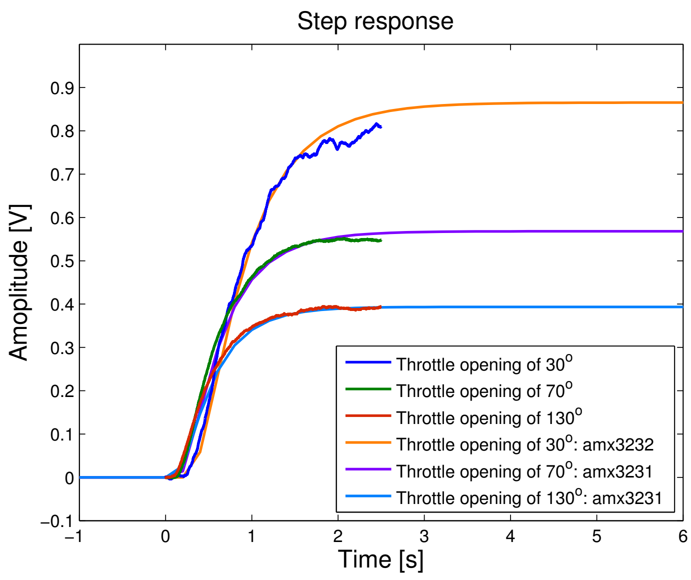


Figure 2.20: Real (from figure 2.15) and simulated step response unit for the best *ARMAX* models calculated.

Considering that a fitness value above 88% is a reasonable approximation and, as such, sufficient to bring a proper system identification, we can assume that the models obtained are acceptable and can be used for further work. Moreover, when using the models obtained in table 2.2, we must consider the offset to have a correct and complete model of the system, for each throttle opening. Furthermore we can change and add models later, if more are needed.

Comparing both models identified with a sample time of $1\,ms$ and $0.2\,s$ in gain, time constant and delay, we find that they are very similar, achieving the objective of having a discrete system model identical to the real continuous one.

If taken in isolation, these models may not be suitable for control, since the dynamics for a low throttle opening can be very different from the one a little below or above, and a controller projected to the first may not work for the second.

2.3 Controller and Supervisor Design Preparation

In the next sections 2.3.1 and 2.3.2, to complete the system identification, we convert the obtained $ARMAX$ model to a discrete state space model, in order to verify controllability and observability characteristics and to design the controller. Furthermore, in order to perform simulations and compare them with to the real process performance, we design a *Local Model Network* (LMN).

2.3.1 Discrete State Space Model

The system under study was identified with a *ARMAX* structure in section 2.2.4 (table 2.2), from where we can obtain a discrete transfer function model for each value of the throttle opening as

$$M \;\Rightarrow\quad y(k) = \frac{B(q)}{A(q)}\,u(k) = \frac{b_1 q^{n-1} + b_2 q^{n-2} + \cdots + b_n}{q^n + a_1 q^{n-1} + \cdots + a_n}\,u(k) \quad , \tag{2.8}$$

where M is the model itself. To represent each of models we use M_j with $j = 1, \ldots, m$ and $m = 3$. We denote the current model by M_i with index $i \in \mathcal{M}$ where $\mathcal{M} = \{1, \ldots, m\}$. The local models M_1, M_2 and M_3 correspond to the throttle openings of $30°$, $70°$ and $130°$, respectively.

Actually, each of these models have a respective offset *b*, so the model in equation (2.8) needs to be rewritten as

$$M \;\Rightarrow\quad y(k) = \frac{B(q)}{A(q)}\,u(k) + b \quad , \tag{2.9}$$

where each offset *b* has the values from *Offset* in table 2.2.

For further work, we need to convert the discrete transfer function from equation (2.9) to a discrete state space model, to have

$$M \;\Rightarrow\quad \begin{cases} x(k+1) &= A'x(k) + B'u(k) \\ y(k) &= C'x(k) + D'u(k) + b \end{cases} \quad , \tag{2.10}$$

where $x \in \mathbb{R}^{n\times 1}$ is the state, $u(k) \in \mathbb{R}$ is the input variable, $y(k) \in \mathbb{R}$ is the measured output and $b \in \mathbb{R}$ is the offset. The matrices of the state space model are $A' \in \mathbb{R}^{n \times n}$, $B' \in \mathbb{R}^{n \times 1}$, $C' \in \mathbb{R}^{1 \times n}$ and $D' \in \mathbb{R}$.

Using the canonic realization, each matrix for each state space model can be determined with

$$A' = \left[\begin{array}{cccc|c} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ \hline \multicolumn{4}{c|}{I_{n-1}} & 0 \end{array}\right] \quad , \quad B' = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad , \tag{2.11}$$

where I_j is a $j \times j$ identity matrix. The matrices of the output equation are

$$C' = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} \quad , \quad D' = 0 \quad . \tag{2.12}$$

Before proceeding to controller design, we check if each model is controllable and observable. Being $\mathcal{C}$ and $\mathcal{O}$ the matrices of controllability and observability such that

$$\mathcal{C} = \begin{bmatrix} B' & A'B' & A'^2B' & \cdots & A'^{n-1}B' \end{bmatrix} \quad , \quad \mathcal{O} = \begin{bmatrix} C' \\ C'A' \\ C'A'^2 \\ \vdots \\ C'A'^{n-1} \end{bmatrix} \quad , \tag{2.13}$$

the system is controllable if n columns of $\mathcal{C}$ are linearly independent and is observable if n rows of $\mathcal{C}$ are also linearly independent, *i.e.*, $rank(\mathcal{C}) = n$ and $rank(\mathcal{C}) = n$.

The models considered are described in the appendix B. All these models are controllable and observable.

2.3.2 Local Model Network design

To be able to perform simulations similar to the real process performance, we need to combine the models obtained in previous work with some weighting function relative to the respective probabilities of each model. This nonlinear model is known as *Local Model Network* (LMN).

Figure 2.21 shows the block diagram of the LMN model, where u is the command input, in volt $[V]$, and $\theta°$ is the throttle opening, in degree. The weights w$_j$ depend on the throttle opening $\theta°$, M_j are the models early obtained and b_j are the output offsets for each model.

As we discussed in section 2.2.4, each model is obtained for a single throttle opening, corresponding to a single flow. Based on this, we defined a probability function in the throttle opening domain for each model, as shows (a) from

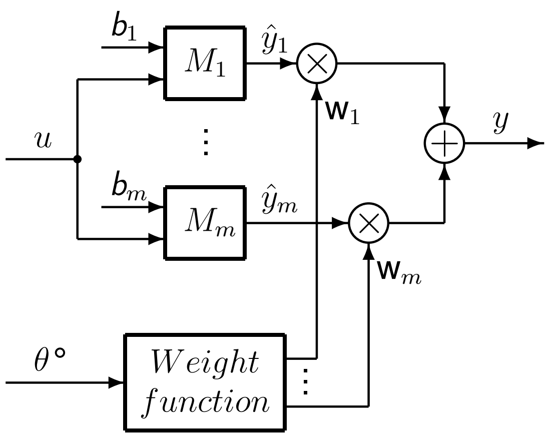


Figure 2.21: Block diagram of the Local Model Network.

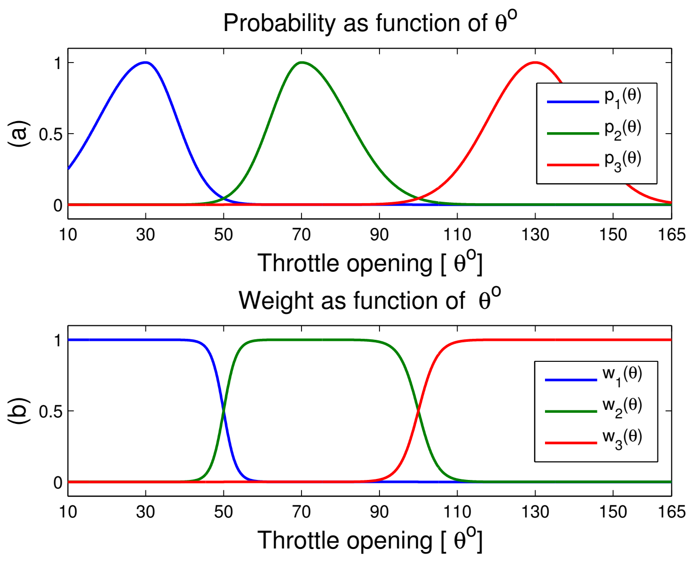


Figure 2.22: Probability (a) and weight (b) signals depending on the throttle opening.

figure 2.22.

The probability functions follow the two-sided Gaussian distribution

$$\begin{cases} p_a(\theta) = e^{-\frac{(\theta-\mu_a)^2}{\sigma_{ab}{}^2}}, & \mu_b < \theta \le \mu_a < \mu_c \\ p_a(\theta) = e^{-\frac{(\theta-\mu_a)^2}{\sigma_{ac}{}^2}}, & \mu_b < \mu_a \le \theta < \mu_c \end{cases}, \tag{2.14}$$

where μ_a is the mean of the Gaussian, σ_{ab} and σ_{ac} are the left and right standard deviations, and $a, b, c \in \mathcal{M}$ are different models. The mean values (μ) are the degrees of each throttle opening $30°$, $70°$ and $130°$. The standard deviations are computed through the distance from consecutive means divided by 5, *i.e.*

$$\sigma_{ab} = \frac{|\mu_a - \mu_b|}{5} = \sigma_{ba} \qquad , \qquad \sigma_{ac} = \frac{|\mu_a - \mu_c|}{5} = \sigma_{ca} \quad . \tag{2.15}$$

The weight is computed by normalizing the probability in plot (a), resulting in plot (b) of figure 2.22, where $\mathrm{w}_j = ||p_1, \ldots, p_m||$. Through equation (2.15), we make sure that the weights have the value of 1 where the probability is 1, *i.e.*, $\mathrm{w}_1(\theta = 30°) = 1$.

For completeness, we also add a low-pass filter, (see appendix D) with $\lambda = 0.95$, to the θ variable in order to simulate the manual manipulation effect. Figure 2.23 shows the throttle opening during the simulation, as well as the weight function results. The simulated process performance can be seen in figure 2.24, where we have the simulated output (magenta) with the estimated models output (blue, green and red).

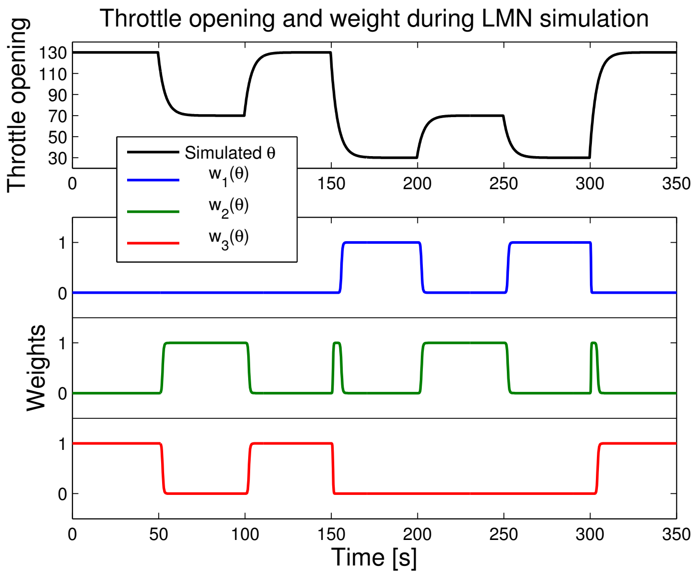


Figure 2.23: Throttle opening (black) and weights values (blue, green and red) during LMN simulation.

The real process performance is shown in figure 2.25, where the throttle opening θ is manually changed each $50\,s$ in the same order as in figure 2.23. We can see that the simulation performance comes close to the real one, even with the low precision of throttle opening manual manipulation.

Despite that, we observe that the overheating of the plant pipe affects the offsets of all models over time. From this experiment we conclude that, for each model, there is a variable offset, caused by the overheating effect. Moreover,

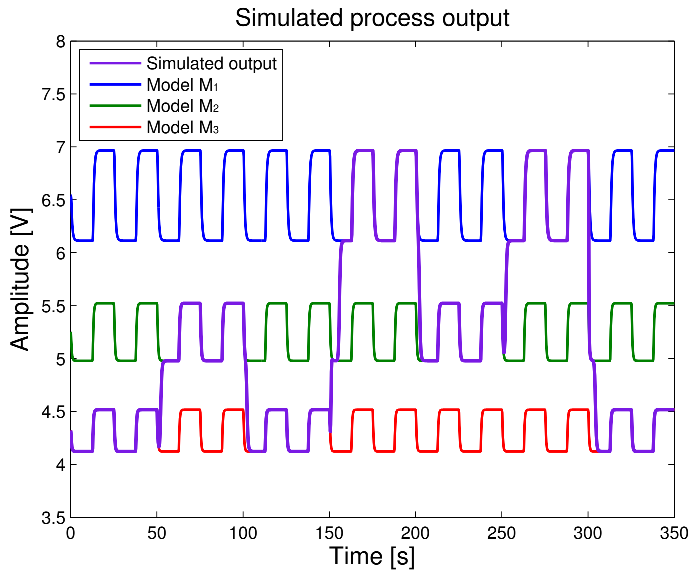


Figure 2.24: Simulated output (magenta) with LMN model and the simulated throttle opening actuation from figure 2.23. Output simulation for the three models in blue, green and red.

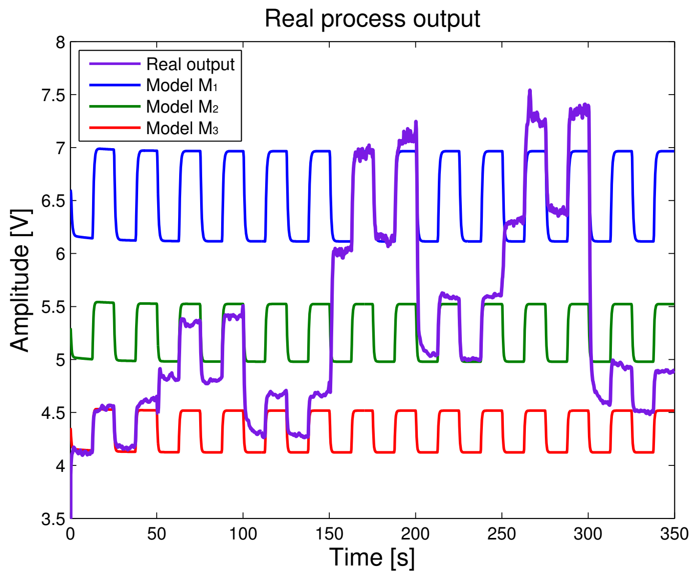


Figure 2.25: Real output (magenta) with throttle opening manual manipulation and output simulation for the three models (blue, green and red).

we may assume that there is a relation between the offsets, which means that when the offset of model M_1 increases, then the offsets of M_2 and M_3 also increases.

We can also observe that, in the last $100\,s$, where the real output goes from M_1 to M_3, the difference between the real output and the corresponding models is nearly the same. Therefore, this led us to guess that the offset variation is equal in every model for each instant in time. Section 2.4 presents a method in order to overcome this problem.

2.4 Estimation of the Offsets

During the experiments with the physical system, we realized that an overheating effect induces to an increase of the output temperature, leading the real output to diverge from the simulated outputs. We think that this effect influences the output offset for the complete throttle opening range. Therefore, we need to analyze how the overheating affects each model and how will we compensate it.

Observing figure 2.25, we focus on how the offset evolves considering the three models separately. Figure 2.26 presents our opinion, where we observe a constant increase of each model offset.

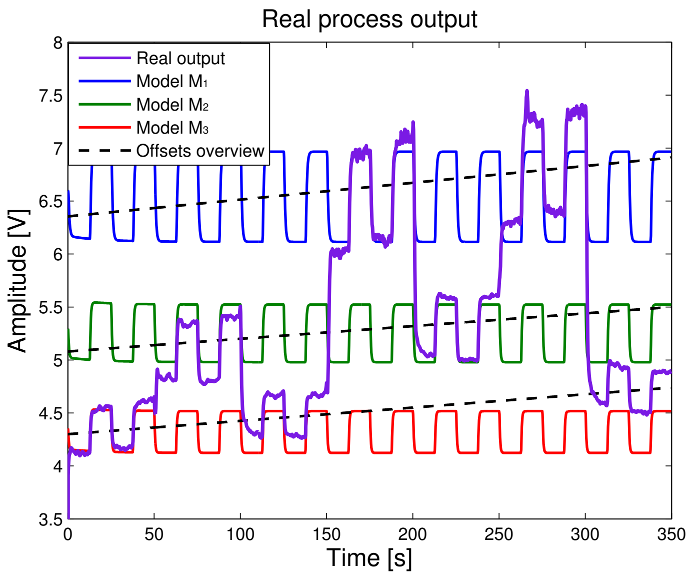


Figure 2.26: Hypothesis of the models offsets (black) respective to the real output, real output (magenta) with throttle opening manual manipulation and output simulation for the three models (blue, green and red).

Therefore we confirm that, not only the overheating affects the offsets of the global system, but also that this influence is similar for the three offsets of the models. Moreover, we can assume that the distance between the models offsets is always constant, even when the overheating happens.

Since the overheating dynamic is slow, we can estimate the offset on-line [2], with

$$\hat{b}_i(k) = \frac{\sum_{j=0}^{k} \alpha_i(j)\beta_i(j)}{\sum_{j=0}^{k} \beta_i^2(j)} \quad , \tag{2.16}$$

where i is the current model index and

$$\alpha_i(k) = -\frac{A_i(q)}{\omega_i(q)} y(k) + \frac{B_i(q)}{\omega_i(q)} u(k) \quad , \quad \beta_i(k) = -\frac{A_i(q)}{\omega_i(q)} u_{-1}(k) \quad , \tag{2.17}$$

with $A_i(q)$ and $B_i(q)$ the numerator and denominator from the transfer function in equation (2.9). We make the polynomial $\omega_i(q) = q^n$, where n is the order of $A_i(q)$, to have a causal transfer function. The $u_{-1}(k)$ denotes the unit step function starting at $k = 0$.

By adding a forgetting factor λ_b to equation (2.16) we get

$$\hat{b}_i(k) = \frac{\sum_{j=0}^{k} \lambda_b^{k-j} \alpha_i(j)\beta_i(j)}{\sum_{j=0}^{k} \lambda_b^{k-j} \beta_i^2(j)} \quad , \tag{2.18}$$

with the value $\lambda_b = 0.4$ chosen. Appendix F shows how to deduce equation (2.18).

Still, equation (2.18) can only calculate one offset, corresponding to the current model, and does not calculate the offsets of the three models needed. However, as we conclude from figure 2.26, the distance between the models is constant, which means that the variance between the current offset and the initial offset is equal for the three models.

Therefore, if we know the initial offsets $\bar{b}_j$ and the current model index i, we are able to compute the offsets of the three models preserving the distance between them. We use the offset values from table 2.2 as the initial offsets $\bar{b}_j$.

Considering this, first we compute the offset variation for the current model as

$$\Delta b_i(k) = \hat{b}_i(k) - \bar{b}_i \quad , \tag{2.19}$$

where $\bar{b}_i$ is the initial offset of the current model M_i. Then, each model offset is calculated with

$$b_j(k) = \bar{b}_j + \Delta b_i(k) \quad , \quad j = 1, \dots, m \quad , \tag{2.20}$$

being $\bar{b}_j$ the initial offsets. We can verify that when $j = i$, the estimated offset is simply $b_i(k) = \hat{b}_i(k)$, but for $j \neq i$, we have $b_j(k) = \bar{b}_j + \hat{b}_i(k) - \bar{b}_i$, where $\bar{b}_j - \bar{b}_i$ is the constant difference between the initial offset of the models.

For example, if we are estimating the offset of the current model M_2, defined as $\hat{b}_2(k)$, we can calculate the offset variation as $\Delta b_2(k) = \hat{b}_2(k) - \bar{b}_2$. We can obtain the current model offset with $b_2(k) = \bar{b}_2(k) + \Delta b_2(k)$, where we can easily see that $b_2(k) = \hat{b}_2(k)$. The offsets of the models M_1 and M_3 are computed as $b_{1,3}(k) = \bar{b}_{1,3}(k) + \Delta b_2(k)$, because the offset variation $\Delta b_2(k)$ is the same in all the models.

When the current model switches, an abrupt change can happen in the offset estimation. To avoid this, we apply a low-pass filter (see appendix D) with $\lambda = 0.98$. Both λ and λ_b were only adjusted after the implementation of the autonomous system in section 3.3. Further in section 3.3, we study the influence of λ in the offset estimation.

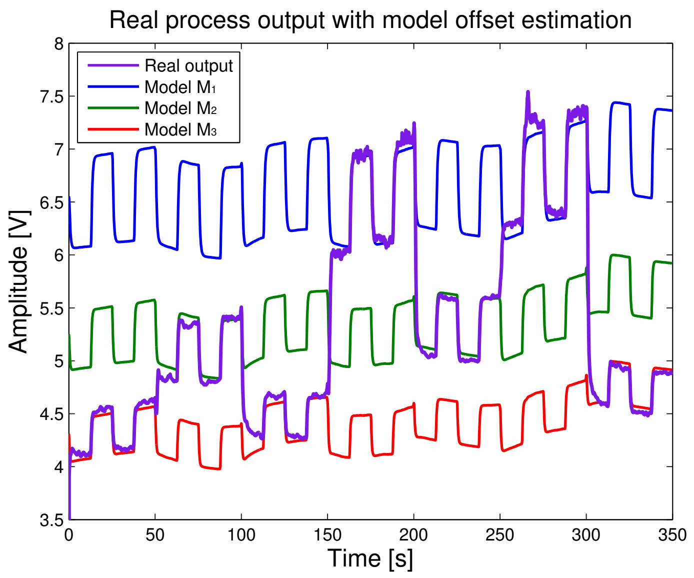


Figure 2.27: Real output (magenta) with throttle opening manual manipulation and output simulation with offset estimation for the three models (blue, green and red).

Figure 2.27 shows the experiment in figure 2.25 with offset estimation, where it is possible to see that the model estimations become closer to the real signal, having a better match during the experiment. More important, the estimation is accurate when the throttle opening changes to another region. However, the model estimation presented in figure 2.27 is only reliable if we know the current model M_i in use.

Chapter 3

Controller and Supervisor Design

3.1 Linear Quadratic Gaussian Controller Design

The type of controllers chosen for this project are the *Linear Quadratic Gaussian* (LQG) controllers. A LQG controller can be obtained with the combination of a *Linear Quadratic Regulator* (LQR), and a *Kalman filter*[1] (KF) that can be individually computed, based on the separation principle [7]. The offset b is discarded in this section.

In this project we intend to use multiple controllers, which will involve controller shifting that can create switching peaks in the command action. To avoid that, we chose to put an integrator after the controller switch [22], leading us to assume the integrator as part of the process, and design the LQG controller to this new process.

Figure 3.1 represents the LQG controller, where the LQR and the KF are shown, together with the process augmented with the integrator mentioned. The switch block will be implemented between the LQG controller and the integrator, on the δu signal (see section 3.3).

The state space model of the process with integrator is obtained from the process state space in equation (2.10), with $b = 0$, and the integrator equation $u(k+1) = u(k) + \delta u(k)$ that leads to

$$\underbrace{\begin{bmatrix} x(k+1) \\ u(k+1) \end{bmatrix}}_{x_a(k+1)} = \underbrace{\begin{bmatrix} A' & B' \\ 0 & 1 \end{bmatrix}}_{A_a} \underbrace{\begin{bmatrix} x(k) \\ u(k) \end{bmatrix}}_{x_a(k)} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{B_a} \delta u(k) \quad , \tag{3.1}$$

[1] Or *Linear Quadratic Estimator* (LQE)

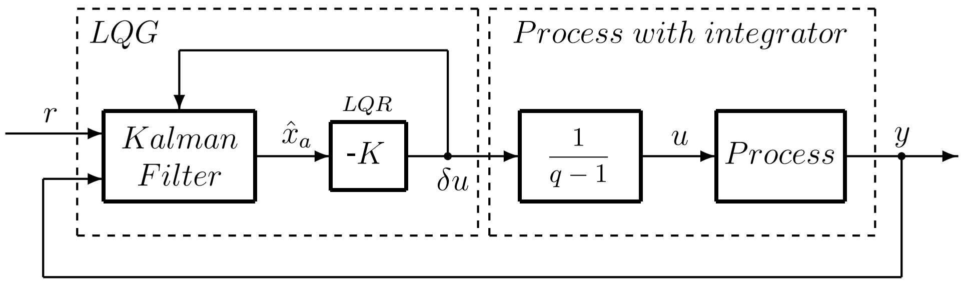


Figure 3.1: System with LQG controller.

with output

$$y(k) = \underbrace{\begin{bmatrix} C' & 0 \end{bmatrix}}_{C_a} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \quad , \tag{3.2}$$

where x_a is the augmented state, A_a, B_a, C_a are the augmented matrices and δu is the incremental command action from LQG controllers.

To design the LQG controller, we assume the plant to be described by the discrete linear time-invariant system given by equations (3.1) and (3.2), including the white Gaussian system noise w and the white Gaussian measurement noise v, resulting in

$$\begin{cases} x_a(k+1) & = & A_a x_a(k) + B_a \delta u(k) + w(k) \\ y(k) & = & C_a x_a(k) + v(k) \end{cases} \quad . \tag{3.3}$$

The quadratic cost function to minimize is

$$J(\delta u) = \lim_{N \to \infty} \sum_{k=0}^{N-1} \left(e^2(k) + R\left[\delta u(k)\right]^2 \right) \quad , \tag{3.4}$$

with $e(k) = r - y(k)$ the tracking error and $R > 0$ a weighting matrix. Since $y^2 = {x_a}^T Q x_a$ it is concluded that $Q = {C_a}^T C_a$. We make $N \to \infty$ to have an infinite-horizon time-invariant LQG problem.

The discrete-time LQG controller is represented by

$$\begin{cases} \hat{x}_a(k+1) & = & A_a \hat{x}_a(k) + B_a \delta u(k) + L\big(\underbrace{y(k) - r}_{-e(k)} - C_a \hat{x}_a(k)\big) \\ \delta u(k) & = & -K \hat{x}_a(k) \end{cases} \quad , \tag{3.5}$$

where the first equation refers to the KF and the second one to the LQR. The observer gain L and regulator gain K are given by

$$\left\{\begin{array}{lcl} L & = & A_a S {C_a}^T (C_a S {C_a}^T + V)^{-1} \\ K & = & ({B_a}^T P B_a + R)^{-1} {B_a}^T P A_a \end{array}\right. \quad , \tag{3.6}$$

which are computed by *Matlab* functions 'dlqr.m' and 'kalman.m' respectively. As the LQG controller has a high overshoot, we can apply the procedure

$$A_a \to \alpha A_a \quad , \quad B_a \to \alpha B_a \quad , \tag{3.7}$$

where $\alpha \geq 1$, that constrains the closed loop poles to be inside a circle of radius $1/\alpha$. This procedure is only used to calculate the regulator gain K. The values for the gains obtained are described in appendix C. The matrices S and P are the solutions of the discrete time *algebraic Riccati equation*

$$\left\{\begin{array}{lcl} S & = & A_a \left(S - S {C_a}^T (C_a S {C_a}^T + V)^{-1} C_a S\right) {A_a}^T + W \\ P & = & {A_a}^T \left(P - P B_a ({B_a}^T P B_a + R)^{-1} {B_a}^T P\right) A_a + Q \end{array}\right. \quad , \tag{3.8}$$

where W and V are respectively the covariance matrices of noise w and v in equation (3.3). In order to apply a loop-transfer recovery procedure, we make $W = q^2 B_a {B_a}^T$ with q a scalar variable that is made to grow.

Solving (3.5), the LQG state space system to be used is

$$C \Rightarrow \left\{\begin{array}{lcl} \hat{x}_a(k+1) & = & \big(A_a - B_a K - L C_a\big)\hat{x}_a(k) - L e(k) \\ \delta u(k) & = & -K \hat{x}_a(k) \end{array}\right. \quad , \tag{3.9}$$

where we define C_i as the LQG controller designed for model M_i.

If the command goes out of the range between $0\,V$ and $10\,V$, it can saturate, causing an integrator windup effect. We can apply an anti-windup block, as figure 3.2 shows, with $K_{sat} = 0.5$, to compensate only the integrator, since it is isolated from the rest of the controller.

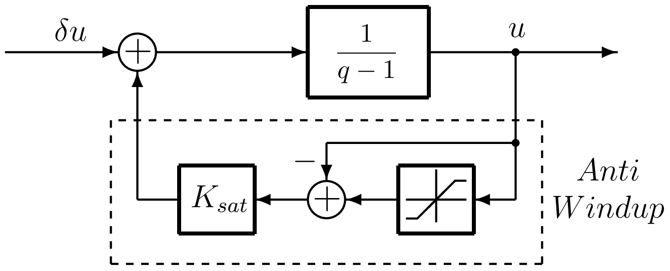


Figure 3.2: Block diagram of the anti-windup implemented to integrator.

For each model M_i obtained, we compute the respective LQG controller C_i with equation (3.9). The values used to design each controller are presented in table 3.1.

Table 3.1: Values for LQG controller design for the three models obtained. The value α is used for the procedure of (3.7), R calculates the LQR gain, and q and V calculate the KF gain.

i	$\theta°$	α	R	q	V
1	30	1.01	6	1	1
2	70	1.02	.5	1	1
3	130	1.005	0.005	1	1

In the experiments shown in figures 3.3, 3.4 and 3.5, we are aiming to analyze the reference tracking performance of each model M_j, actuated by its respective LQG controller C_j, where $j = 1, \ldots, m$. Each figure shows three experiments, each one is the signal corresponding to a controller C_i that matches model M_i.

Figure 3.3 shows three different simulations with the simulated reference tracking performance for each model, around the corresponding equilibrium point, controlled by the respective LQG controller. In figure 3.3 the results of three different simulations are plotted superimposed. Each of the simulations is obtained with a set-point that stays near the operating point where each of the local models is valid. Figure 3.4 shows the respective performance in the actual physical system. The plot shows the data of three different experiments.

Although we obtained similar results in the output signals, there is an increasing divergence in the command action between figures 3.3 and 3.4. This difference can be explained by the overheating mentioned before. As the temperature increases, the command action of C_1 and C_2 must decrease in order to reach the same output. Moreover, we verify that C_3, in this case, is not affected by the overheating.

For global experiments we choose, for the reference, a square signal with mean of $7\,V$ and an amplitude of $0.4\,V$, where ideally no controller will saturate. Figure 3.5 shows the real reference tracking performance experiments, as figure 3.4, but this time around the operating point of $7\,V$.

We confirm that the overheating is affecting the command action of C_1, C_2 and also of C_3 unlike figure 3.4. By this experiment, we state that the overheating affects the output of every model around the chosen operating point of $7\,V$.

Moreover, we performed an experiment to find out which controller has the best performance, when different throttle openings are used. For that, we performed three experiments, each one consisting on using a fixed controller. Dur-

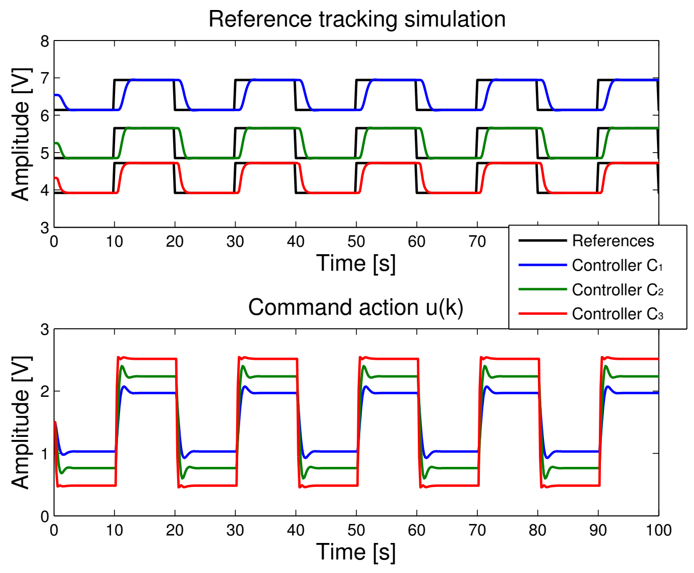


Figure 3.3: Simulated reference tracking performance around the equilibrium point of each model.

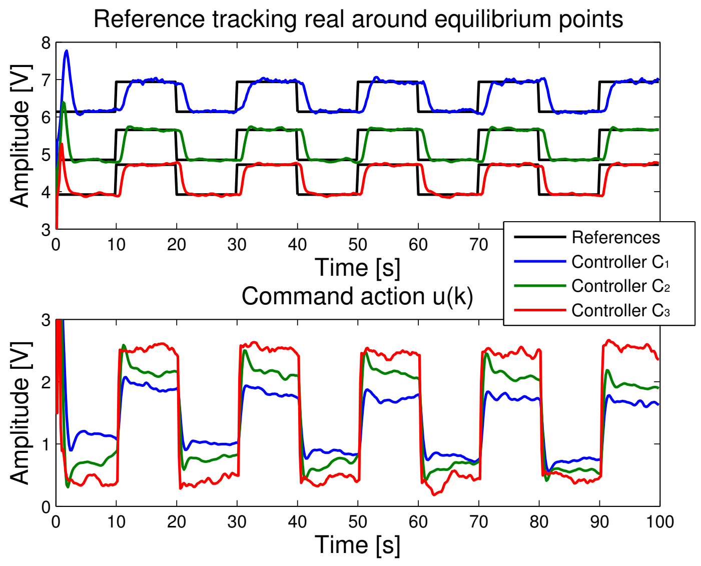


Figure 3.4: Real reference tracking performance around the equilibrium point of each model.

ing each experiment the throttle opening is changed manually, following the order $130°$, $70°$ and $30°$ each $50\,s$. Figure 3.6 presents the three experiments performed corresponding to the three controllers designed.

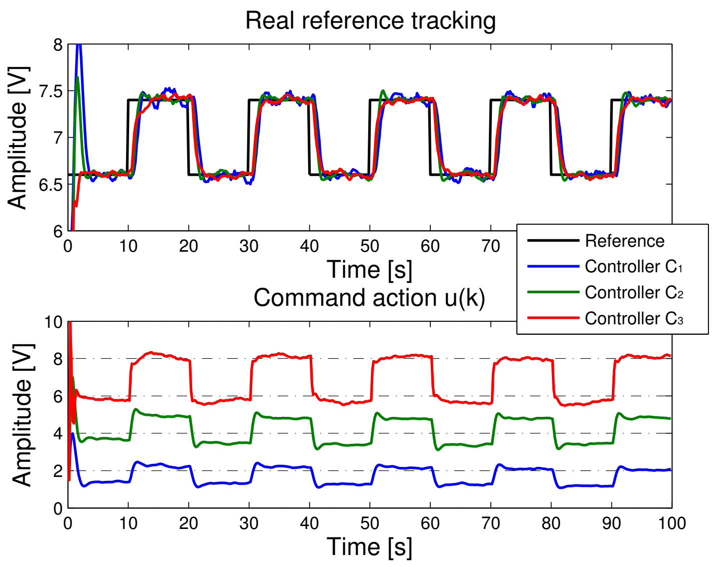


Figure 3.5: Real reference tracking performance around the operating point of $7\,V$.

We observe that one controller has good performance when θ matches the throttle opening for its corresponding model. The best controller would be C_2, but still, for any controller, the performance decreases whenever θ does not match the model for which it was designed.

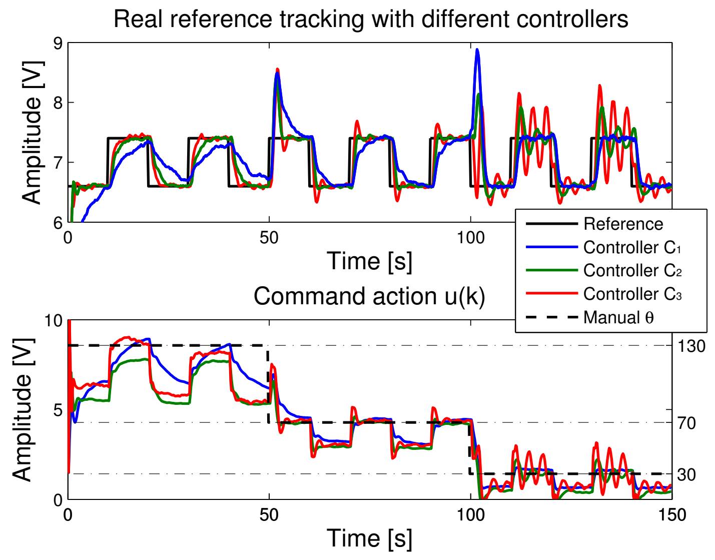


Figure 3.6: Three experiments where the throttle opening changes and in which, only one fixed controller is used.

Since none of the controllers achieved an acceptable performance, we conclude that one fixed controller is insufficient to control the system in an extended operating range. Therefore, we need a controller that matches the model valid in the current operating region. To use the current model controller C_i, we need to identify in which operating region the system is actually working, a task performed by the supervisor. Then we select the corresponding controller with the switch already mentioned. In order to ensure stability, a dwell-time switching logic is used.

3.2 Supervisor Design

To implement a supervisor able to detect the model that best fits the current process dynamic behavior, we can deal with two characteristics. First, we can compute the error between the estimated models output and the measured plant output, due to distinct offsets. Second, we can compare the performance of the transient responses between the models and the measured plant output. For each one we designed a supervisor and then we implemented a logic switch to achieve a better performance.

3.2.1 Offset supervisor

We take advantage of the offset in the system to have faster model detection. The offset supervisor has the function of calculating which model is 'closest' to the real simulation, which means that we need to compute the error between the real signal and the simulated signals, where the offsets are included.

The offset supervisor is shown in figure 3.7. First we estimate outputs $\hat{y}_j$ with the command action u and the models M_j, as we did in LMN, adding the estimated offset b_j. After we get the error, by subtracting the estimated outputs with the real one y, we compute the quadratic error

$$w(k) = |e|^2 = |\hat{y}(k) - y(k)|^2 \quad . \tag{3.10}$$

The performance signal is defined as

$$\pi_o(k) = \lambda_o \, \pi_o(k-1) + (1 - \lambda_o) \, w(k) \quad , \tag{3.11}$$

representing a low-pass filter where $\lambda_o = 0.6$.

The logic switch block finds the performance signal with lower value and returns its index σ_o

$$\sigma_o(k) = \underset{j}{argmin} \; \pi_{o,j}(k) \quad , \tag{3.12}$$

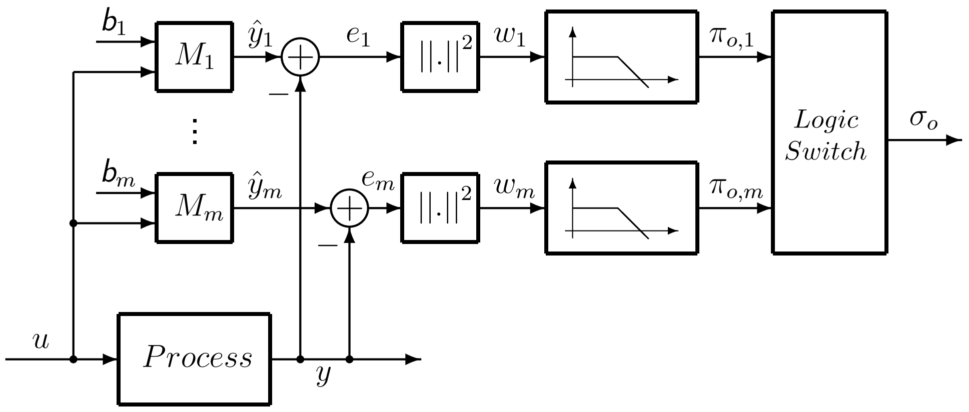


Figure 3.7: Diagram of supervisor block with offset compensation.

where $\sigma_o \in \mathcal{M}$ is the resulting detection of the offset supervisor.

If the supervisor model detection fails and gives an incorrect index i, then the offset supervisor becomes useless. Further in section 3.2.2, we include the transient supervisor to rectify the results of the offset supervisor when needed, to overcome this issue.

3.2.2 Transient supervisor

The offset supervisor detection may fail if the current model identification is not correct, occurring unpredictable results from that. Because the transient supervisor does not depend on the current model identification, it is more accurate than the offset supervisor, but only if the reference changes, and the throttle opening maintains the same.

We use the transient supervisor for its accurateness if by any chance the offset detection fails. Here the offset is discarded, which means that $b_j = 0$ for $j = 1, \ldots, m$, and we focus in comparing the transient responses. We can foretell that this supervisor will be only useful when a change in the reference r occurs, because that is when a transient may appear.

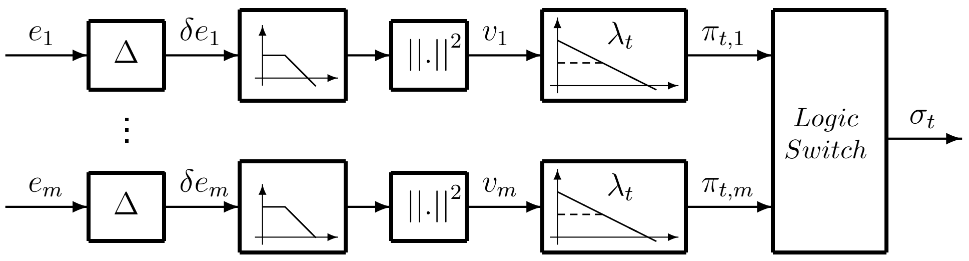


Figure 3.8: Block diagram of the transient supervisor.

Figure 3.8 shows the transient supervisor process. We compute the errors e_j like in figure 3.7, apart from the offset $b_j = 0$. Then we derive the error

$$\delta e(k) = e(k) - e(k-1) \quad , \tag{3.13}$$

and filter $\delta e(k)$ with a low-pass filter (see appendix D) with $\lambda = 0.9$, to eliminate high frequency noise. After we compute the quadratic error v_m, the performance index is given by

$$\pi_t(k) = \lambda_t \, \pi_t(k-1) + v(k) \quad , \tag{3.14}$$

where λ_t is a variable forgetting factor [13]. The reason for having a variable forgetting factor comes with the necessity of forgetting past samples when a change in r or y is detected. When only y changes, it means that the throttle opening is switching and when r changes, it means that a transient is about to happen. In both situations it is useful to forget past values so, we consider that a change happens if $|\delta r| + |\delta y| > \delta_{ry}$, with the threshold of $\delta_{ry} = 0.05\,V$.

Therefore, when no changes occur, we use $\lambda_t = 1$ and all the past samples weigh in the performance evaluation. When a change occurs, we use $\lambda_t = 0.3$ to forget past samples, giving more weight to the new ones. Appendix E shows the implementation of the variable forgetting factor.

The logic switch block is the same as in the offset supervisor

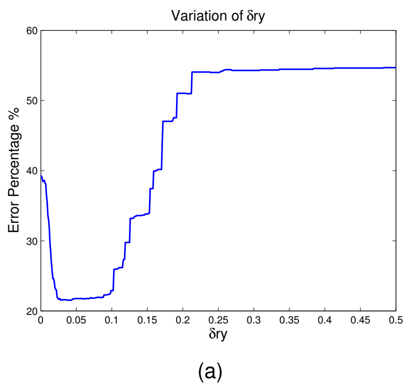
$$\sigma_t(k) = \underset{j}{argmin} \; \pi_{t,j}(k) \quad , \tag{3.15}$$

where $\sigma_t \in \mathcal{M}$ is the resulting detection of the transient supervisor. In figure 3.9 we analyze the influence of the variation of the threshold δ_{ry} in the plot 3.9a and the influence of the variation of λ_t in the plot 3.9b. The error percentage is related with the number of failed detections made by the transient supervisor.

From figures in 3.9, we see a possible range of values in which we can work with for each variable. When we vary the δ_{ry}, we are looking to for the threshold that tells when to change λ_t. On the other hand, when we vary λ_t, we are looking for the forgetting factor that tells how many past samples are used. Later, we could optimize the transient offset estimation by finding the best values for each variable to achieve the minimal error.

Figure 3.10 shows the result of an experiment that consists in manually switching to the right controller, after manually changing the throttle opening, in the same order of figure 2.23. We can observe in figure 3.11 the detections of the offset and transient supervisors.

As we mentioned before, the offset supervisor switches quickly and the



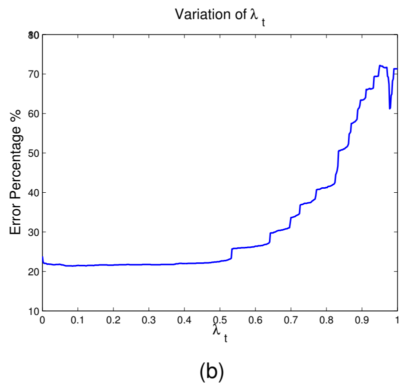


Figure 3.9: Influences of the threshold δ_{ry} (a) and the forgetting factor λ_t (b) on the error of the transient supervisor detection.

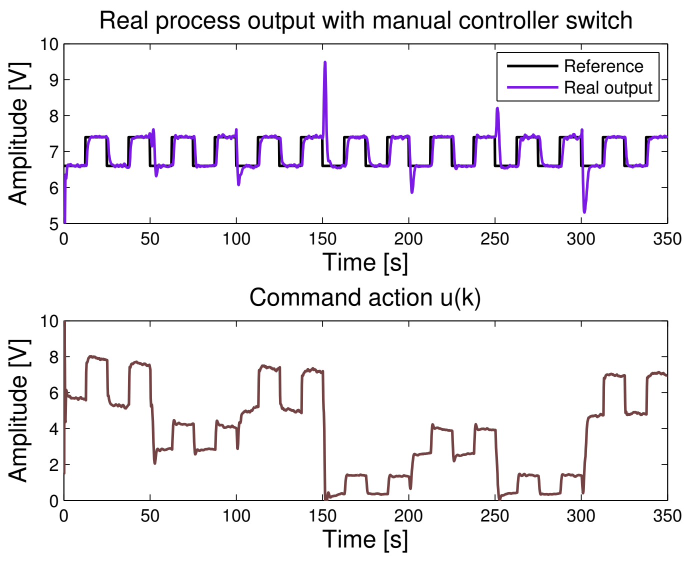


Figure 3.10: Reference tracking real output (magenta) with manual throttle opening and controller switch.

transient supervisor needs a change in the reference to rectify its detection. We can also note that after the throttle opening changes, the detection of the transient supervisor fails until the reference changes, only then detecting the correct index.

Furthermore, we note two outliers in σ_t, happening when $\sigma_o = 1$ that could be related to the effect of the overheating in the dynamic. In the first $20\,s$ of simulation the offset estimation also fails due to unknown initial conditions.

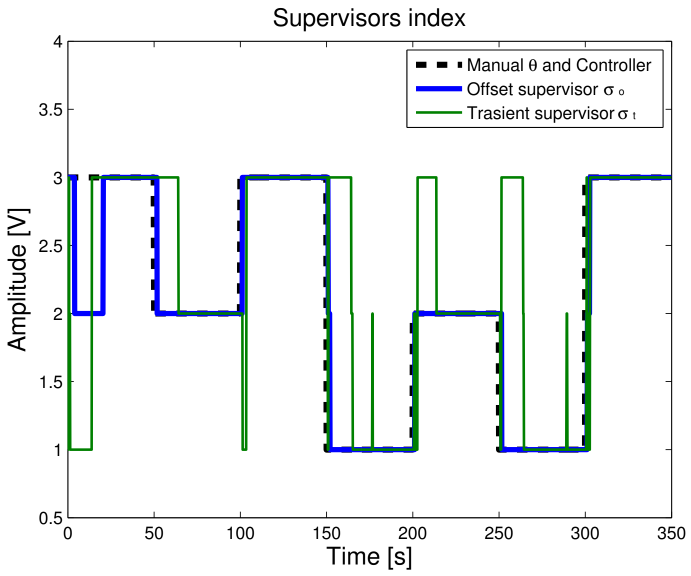


Figure 3.11: Offset and transient supervisors index from the manual switch controller experiment output in figure 3.10.

3.2.3 Supervisor logic design

Thereafter, we develop a logic circuit to take advantage from both the supervisors described before. The logic switch for the resulting index $\sigma = f(\sigma_o, \sigma_t)$ follows the *SR flip-flop* circuit [23]. The *SR flip-flop* circuit has S and R as inputs, Q as output and follows the "set-reset" behavior, where the output Q is set to 1 if $S = 1$ and reset to 0 if $R = 1$. When both S and R are 0 no change happens and the state where both are 1 must be avoided.

Table 3.2 presents the logic from Δry, $\Delta\sigma_o$ and $\Delta\sigma_t$ to S and R, from S and R to Q and then, in the last line, we assign to the supervisor output, σ, the value of σ_o or σ_t according to the logical result of Q. In this case, we choose $\Delta ry = 1$ when $|\delta r| + |\delta y| > 0.5\,V$, $\Delta\sigma_o = 1$ when $\delta\sigma_o \neq 0$ and $\Delta\sigma_t = 1$ when $\delta\sigma_t \neq 0$. The supervisor output is defined as σ and is shown in the last line of the table. The positive logic of Δry and $\Delta\sigma_t$ makes $\sigma = \sigma_t$ and $\Delta\sigma_o$ makes $\sigma = \sigma_o$. In order to avoid both $S = 1$ and $R = 1$, we use priorities in the way that, the highest priority forces lower priorities to 0.

We give priority when $\Delta ry = 1$, setting $\sigma = \sigma_t$. Then, whenever the offset supervisor index σ_o changes ($\Delta\sigma_o = 1$), we make $\sigma = \sigma_o$ because that is when the transient detection fails. If no changes are detected neither in Δry or in $\Delta\sigma_o$, then if $\Delta\sigma_t = 1$ we set $\sigma = \sigma_t$.

The logic mentioned is not sufficient to have a good performance in the supervisor output and we make use of timers. Therefore, we use an *on-delay*

Table 3.2: Supervisor logic table to choose the supervisor index to be used. The ‘x’ means the supervisor output holds its previous value.

Δ_{ry}	0	1	0	1	0	1	0	1
$\Delta\sigma_t$	0	0	1	1	0	0	1	1
$\Delta\sigma_o$	0	0	0	0	1	1	1	1
SR flip-flop								
S	0	1	1	1	0	1	0	1
R	0	0	0	0	1	0	1	0
Q	x	1	1	1	0	1	0	1
Supervisors index								
σ	x	σ_t	σ_t	σ_t	σ_o	σ_t	σ_o	σ_t

timer with 15 samples delay to rectify the transient supervisor outlier, which is implemented to σ when $Q = 1$. We make $\sigma = \sigma_o$ last longer with a *monostable timer* with 30 samples delay applied to $\Delta\sigma_o$. In the end, we apply to σ a *monostable timer* with 10 samples delay. This timer is known as *dwell-time* and forces a controller to be active for a minimum period of time, in this case $\tau_D = 2\,s$. In this case, it also helps to correct the supervisor output.

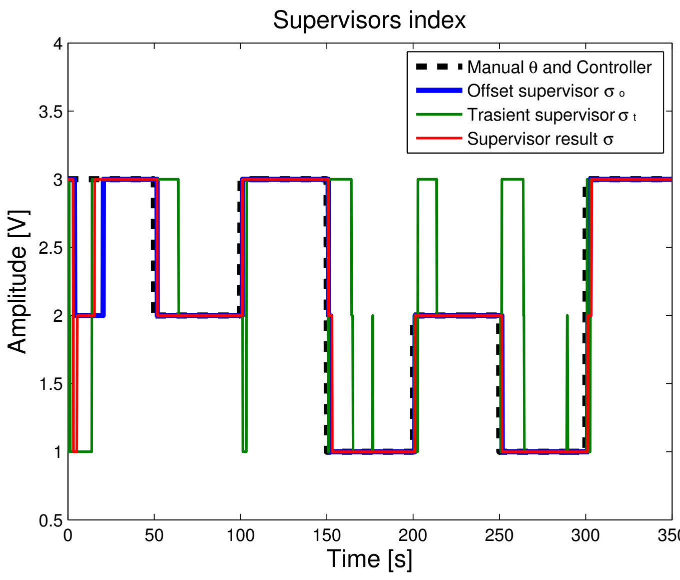


Figure 3.12: Offset, transient supervisors index and supervisor output from the manual switch controller experiment output in figure 3.10.

Figure 3.12 shows the supervisor output. We can observe that most of time the $\sigma = \sigma_o$ and only in the beginning appears $\sigma = \sigma_t$ delayed. We can also verify that, in the first $20\,s$, σ_o is rectified only after σ_t.

3.3 Global Autonomous System and Results

Combining the LQG controllers and supervisor designed, we have an autonomous system that is alerted to the changing of the throttle opening, or flow, and is able to change its controllers in order the have the best performance in the output.

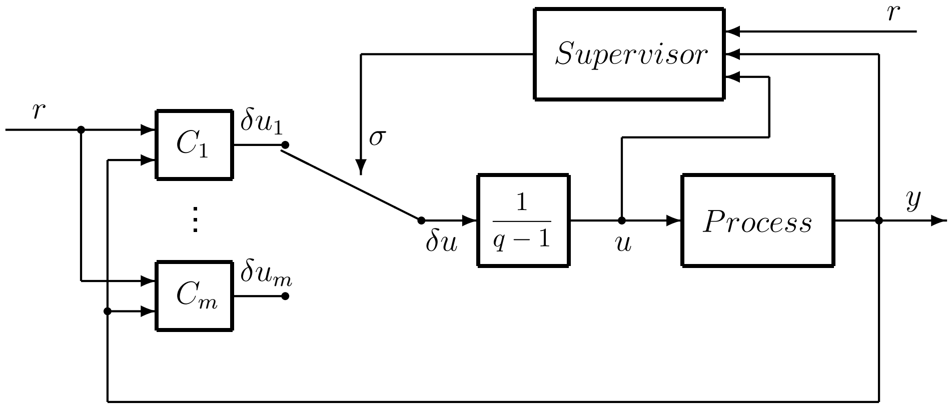


Figure 3.13: Block diagram of the global control system.

The complete system is represented in figure 3.13, where the blocks C_1 to C_m represent the LQG controllers from figure 3.1 and the *Supervisor* block contains the offset and transient supervisors plus the logic circuit. We feedback the supervisor output, making $i = \sigma$, so that the controller C_i becomes C_σ.

Now that we are able use the autonomous system, we can see in figure 3.14 how low-pass filter of the offset estimation influences the system performance. The error percentage consists on the error between the reference and system output. Here, we observe that only values of lambda close to 1 causes less error in the performance, because that is when the offset estimation is more constant.

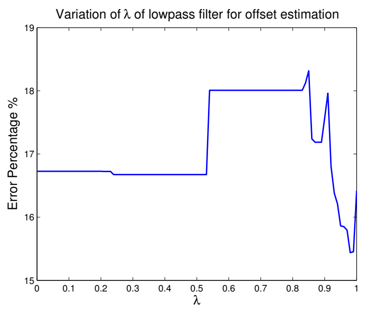


Figure 3.14: Influences of the low-pass filter parameter λ of the offset estimation in the error between the reference and the system output.

We can also compare the performance of the supervisors designed, first separately, and then combined. Figure 3.15 presents the experiment results for reference tracking with the real output signal and the estimated models outputs. Figure 3.16 presents the real output signal and the supervisor detection index when using the offset supervisor. The same goes for 3.17 and 3.18 when using the transient supervisor and with 3.19 and 3.20 when combining both supervisors.

Figures 3.15, 3.17 and 3.19 shows the same experiment made in figure 3.10, but now the system is completely autonomous. We can observe that the output is stable and tracks the reference like in section 3.1. However, the models estimation differs between this figures, which lead us to analyze the supervisors performance.

Analyzing the supervisor index plot in figures 3.16, 3.18 and 3.20, we can see that the best fit is when both supervisors are used. When the throttle opening changes, the regulation of the supervisor index takes a maximum of $16.6\,s$ using the transient and $3\,s$ using both supervisors. With the offset supervisor the same does not happen, where the supervisor index is wrong until $150\,s$ and from there it is similar to using both supervisor. Moreover, other experiments made using the offset supervisor shows that its initial detection is random and, when the first index matches the current model, the supervisor performance becomes similar to using both supervisors.

We can observe that, after each throttle opening changes, the output performance decreases when using the transient supervisor comparing to when using both combined supervisors. Furthermore, we can see that when the throttle opening changes, to the output follow back tracking the reference, it takes a maximum of $10\,s$ using the transient supervisor and $6\,s$ using both supervisors.

From the supervisors performance analysis, we conclude that using both supervisors is the best choice. If we have a good supervisor performance, then we should have a good reference tracking performance.

Moreover, looking just for the results of the experiment using both supervisors, in figure 3.20, we can see that it takes between $0.4\,s$ to $3\,s$ for the supervisor to change to the correct model index. Apart from that, there is a misidentification in the first $30\,s$, which can happen due to initial conditions of the supervisors.

In addition, there are some outliers in transient supervisor detection, both in figures 3.12 and 3.20. We presume there are two reasons for this to happen. The first reason can be the saturation of the command control that directly affects the real output transient. The second reason may be a divergence in the dynamic behavior caused by the overheating effect.

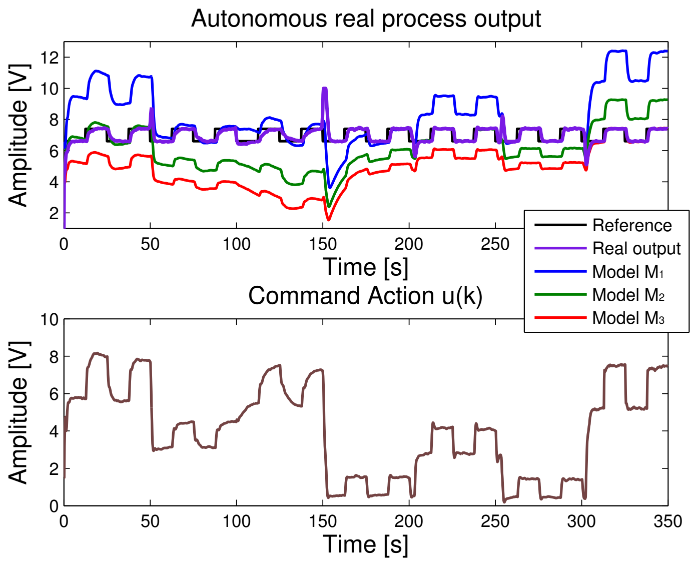


Figure 3.15: Autonomous reference tracking performance using only the offset supervisor.

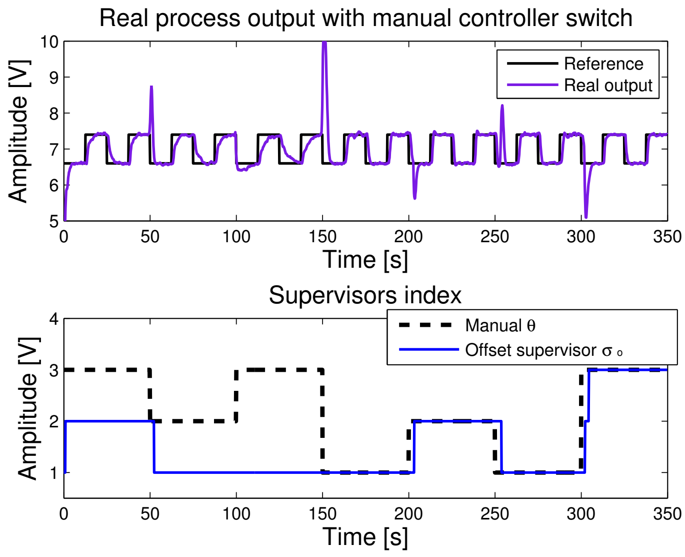


Figure 3.16: Offset supervisor index and supervisor output from the autonomous real process output.

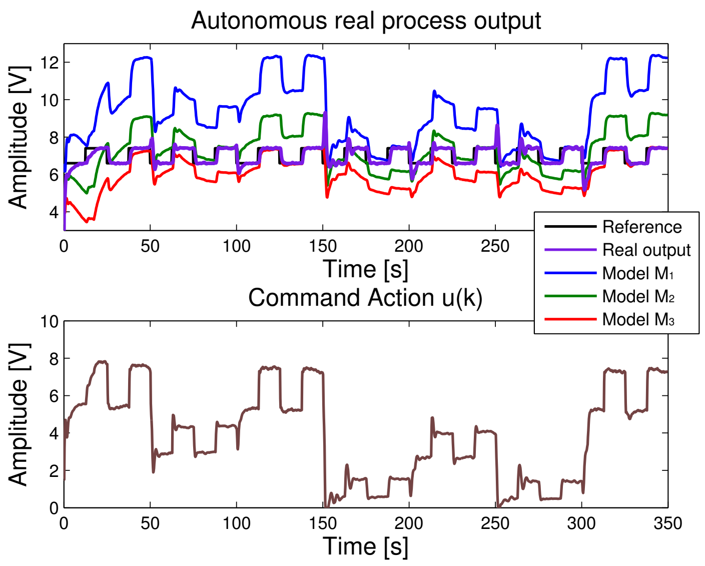


Figure 3.17: Autonomous reference tracking performance using only the transient supervisor.

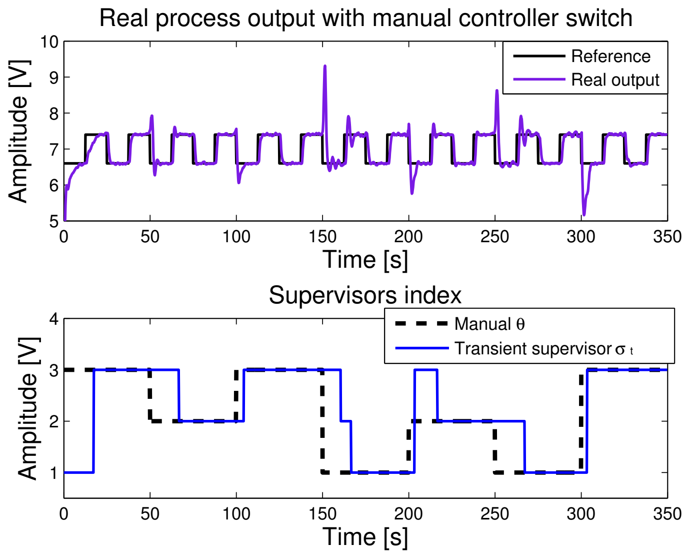


Figure 3.18: Transient supervisor index and supervisor output from the autonomous real process output.

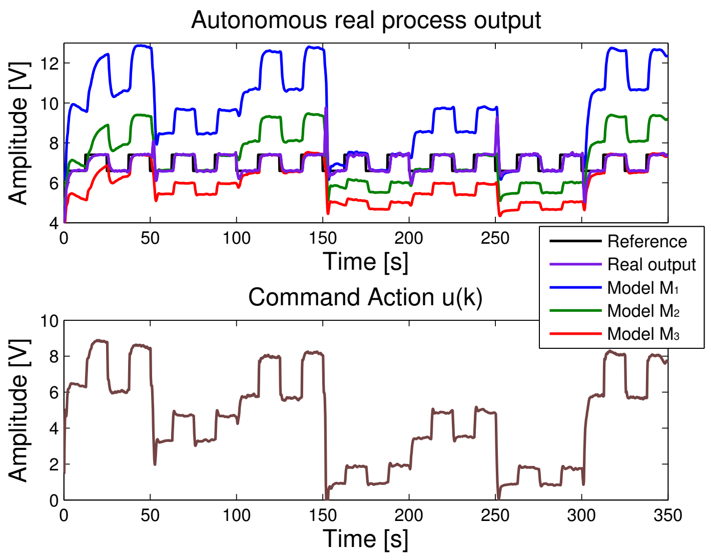


Figure 3.19: Autonomous reference tracking performance using both supervisors.

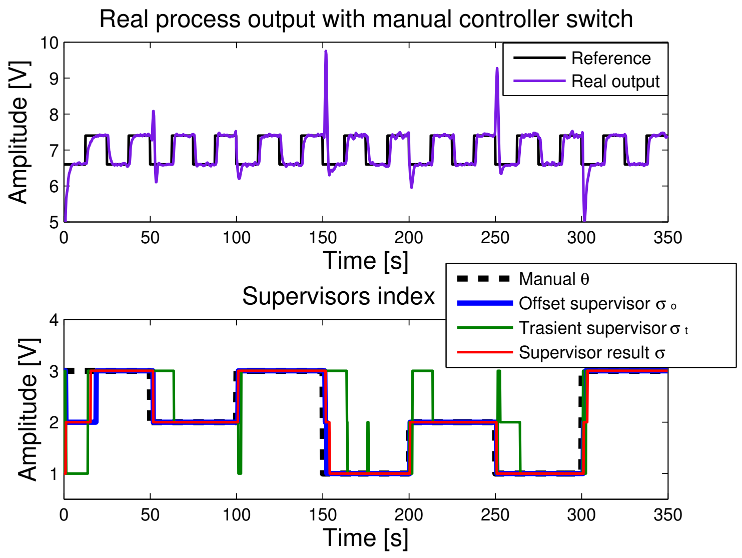


Figure 3.20: Offset, transient supervisors index and supervisor output for the autonomous real process output.

Chapter 4

Adaptive Control

The dynamic of some systems can change over time, which can be caused by external or internal factors. In the system in study, these factors can be room temperature change, overheating of the plant pipe or the saturation of the command action. In these situations, both controller and supervisor may become inadequate to achieve a good performance of the system.

We use adaptive control to estimate the new dynamic and then update the controller and supervisor, in order to maintain a good system performance. We can use the *Recursive Least Squares* (RLS) with forgetting factor algorithm to estimate the new dynamic parameters. Then we can design the LQG controller with the estimated values obtained.

4.1 Recursive Least Squares

The *Recursive Least Squares* (RLS) is an algorithm that uses past estimated values to estimate the new model parameters [20]. First we show how to get to the matrix notation and then we present the algorithm.

Due to the use of the LQG controller, we can consider the noise of the system as uncorrelated and, instead of using the ARMAX structure, we represent the system dynamic with the ARX structure

$$\begin{aligned} &y(k) + a_1\, y(k-1) + \ldots + a_{n_a}\, y(k-n_a) = \\ &\qquad = b_1\, u(k-n_k) + \ldots + b_{n_b}\, u(k-n_k-n_b+1) \; + e(k) \quad . \end{aligned} \tag{4.1}$$

To estimate the parameters of the system, we first need to write the model as

$$y(k) = \varphi^T(k-1)\theta(k) + e(k) \quad , \tag{4.2}$$

where we define the regressor φ as

$$\varphi(k-1) = \big[-y(k-1)\ \dots -y(k-n_a)\ \ u(k-n_k)\ \dots\ u(k-n_k-n_b+1)\big]^T , \quad (4.3)$$

and the parameters vector is

$$\theta = \big[a_1\ \dots\ a_{n_a}\ b_1\ \dots\ b_{n_b}\big]^T \quad . \quad (4.4)$$

The linear least squares cost function is minimized by

$$J\big(\theta(k)\big) = \sum_{i=0}^{k} \lambda_{RLS}^{k-i}\big[y(i) - \varphi^T(i-1)\theta(k)\big] \quad , \quad (4.5)$$

where λ_{RLS} is the forgetting factor. The RLS algorithm is computed with

$$\begin{cases} K_{RLS}(k) = \dfrac{P(k-1)\varphi(k-1)}{\lambda_{RLS} + \varphi^T(k-1)P(k-1)\varphi(k-1)} \\ \hat{\theta}(k) = \hat{\theta}(k-1) + K_{RLS}(k)\big[y(k) - \varphi^T(k-1)\hat{\theta}(k-1)\big] \\ P(k) = [I - K_{RLS}(k)\varphi^T(k-1)]P(k-1)/\lambda_{RLS} \end{cases} , \quad (4.6)$$

where K_{RLS} is a *gain vector*, P denote the inverse of the autocorrelation matrix and $\hat{\theta}$ is the estimated parameters vector.

In this project, we initialize $\hat{\theta}(-1) = \theta$ where θ is the *a priori* parameters from *ARMAX* structure in table 2.2, and $P(-1) = 10^{-4}I$ that means that we are certain of the initial values. If the initial parameters $\hat{\theta}(-1)$ were unknown, we would rather use $P(-1) \geq 10^2 I$. The forgetting factor used is $\lambda_{RLS} = 0.99$, in order to have a smooth convergence.

Next, we demonstrate an example of the RLS algorithm performance. In the simulation of figure 4.1, we have M_2 representing the real model and M_3 representing the simulated model. The parameters estimation starts at $20\,s$ and, thenceforth, we update the simulated model every $10\,s$. In section 4.2 we explain when we estimate parameters (E_p) and when they are updated (U_p).

Figure 4.1 shows the simulation results, where the expected output signal (green) is the result of the regressor vector times the current estimated parameters, $y_{es}(k) = \varphi^T(k-1)\hat{\theta}(k)$.

We can observe that the RLS algorithm guarantees the convergence of the simulated model to the real model and it can be implemented in the autonomous system.

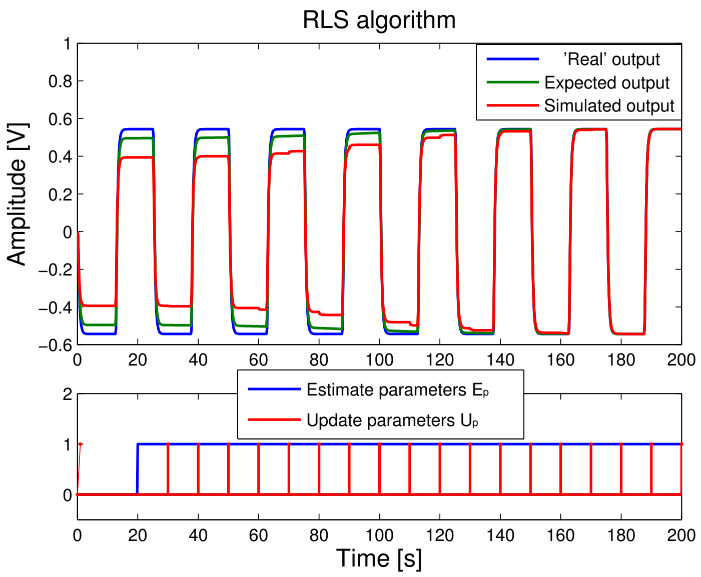


Figure 4.1: Example of the performance of the RLS algorithm.

4.2 Adaptive Control Implementation

We can implement Adaptive Control in the system of figure 3.13 of section 3.3 by adding the blocks of figure 4.2. The RLS algorithm must have the current system output, without the offset, *i.e.* $y - b_i$, the input and output past values φ and the current model index σ and returns the estimated parameters vector $\hat{\theta}$.

The *Update* block updates the current model state space matrices A_i and C_i[1] with the estimated parameters $\hat{\theta}$. The blocks that we can update are the models of the supervisor M, the estimated offsets b and we can redesign the LQG controllers C. The RLS algorithm is active when $E_p = 1$ and update block is enabled only when $U_p = 1$.

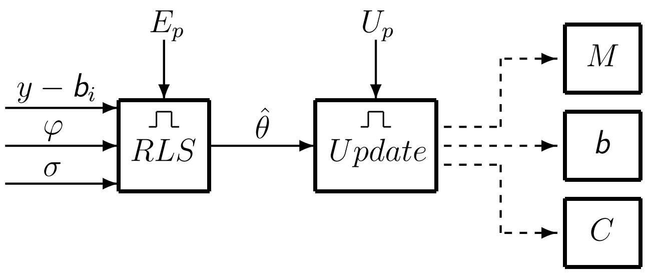


Figure 4.2: Diagram block for implementation of Adaptive Control in the system of figure 3.13.

[1] The state space matrixes B_i and D_i are constant.

As we are dealing with a multiple model system, we need to choose when it is reliable to estimate the new parameters and also, when to update the supervisor and LQG controllers.

Considering the supervisor index on figure 3.20, we assume that the correct model identification happens when all three signals of the figure (σ_o, σ_t and σ) are the same. Therefore, this is when we can estimate the new model with the estimated coefficients and when we make $E_p = 1$. Still, we anticipate some corrections of when to make the estimation.

Having the regressor φ as a vector with input and output past values, where its oldest values are $y(k - n_a)$ and $u(k - n_k - n_b + 1)$, the maximum delay corresponds to $n_\varphi = max(n_a, n_k + n_b - 1)$. Therefore, we can only start the estimation n_φ samples after the correct model identification begins. Based on the models computed in table 2.2, the maximum delay is $n_\varphi = 3$. We choose to update the parameters, making $U_p = 1$, every $10\,s$ after the estimation begins, which corresponds to 50 samples, with the sample time of $h_s = 0.2\,s$.

However, the update does not happen every time $U_p = 1$ because it also depends on the parameters of the estimated model. To approve the update, we need to compute the distance between the original model and the estimated model and then, compare this distance with some thresholds. We use as distance the Vinnicombe's variation, defined by the gap metric between two systems, denoted by $\delta_\nu(S_i, S_j)$. This distance is called the ν-gap metric and can be computed by the 'gapmetric.m' function in *Matlab*[2] [24].

Moreover, we define an upper threshold δ_{up} and a lower threshold δ_{lo}. The reason for having an upper threshold δ_{up} is to avoid the estimated model to get too far from its original model and become similar to the next model. If the estimated model is slightly different from the original one, we prefer not to update the parameters, thereby having a lower threshold δ_{lo}, to maintain the original parameters. Summarizing, we only update the estimated model if $U_p = 1$ and then if $\delta_{lo} < \delta_\nu(S_i, S_j) < \delta_{up}$, where S_i denotes the original continuous model and S_j is the estimated continuous model. Table 4.1 presents the distance between each original model and the closest one and the values used for the lower and upper thresholds. We define $\delta_{\nu_{min}} = min(\delta_\nu(M_i, M_{i-1}), \delta_\nu(M_i, M_{i+1}))$ and $\delta_u = \delta_{\nu_{min}}/2$.

We previously state that, after the throttle opening changes, it takes a maximum of $3\,s$ to the supervisor output switches to the correct one. Therefore, the last acceptable estimated parameters that we must use to perform the update are delayed by 15 samples.

Having the controllers changing during adaptive control simulation, it would

[2] The ν-gap is designed to continuous systems. A conversion of the discrete models to continuous time models is made using the 'd2c.m' function of *Matlab*.

Table 4.1: Vinnicombe's variation and ν-gap metric thresholds.

Model	$\delta_{\nu_{min}}$	δ_l	δ_u
1	0.2093	0.01	0.1047
2	0.1225	0.01	0.0612
3	0.1225	0.01	0.0612

be interesting to verify the controllers gains. We can compute the controllers gains with the controller transfer function converted from the state space model from 3.9 to

$$C(q) = \frac{b_1^C q^{n-1} + b_2^C q^{n-2} + \cdots + b_n^C}{q^n + a_1^C q^{n-1} + \cdots + a_n^C} \quad . \tag{4.7}$$

The stationary gain is computed by making $q \to 1$ in (4.7)

$$C(1) = \frac{b_1^C + b_2^C + \cdots + b_n^C}{1 + a_1^C + \cdots + a_n^C} \quad . \tag{4.8}$$

In this part of the work, we are not able to change the parameters during real time experiments. Therefore, we will only present the performed simulations.

Figure 4.3 shows the adaptive control simulation, but without any update due to the thresholds specified. The simulated throttle opening and supervisor output for this simulation are in figure 4.4, where we also present the controllers gains.

With this simulation, we are able to tell that the parameters update does not affect the system performance when we are dealing with the correct number of outputs (and the correct throttle openings). However, we need to test how the adaptive control improves the performance when one model is removed.

Therefore, in the second simulation, we removed the model M_3, and tried the same throttle opening change sequence. Figures 4.5 and 4.6 presents the simulation without adaptive control to compare with the simulation which use adaptive control from figures 4.7 and 4.8.

For the third simulation we do the same simulation, but this time removing the model M_2 and we present the results in figures 4.9, 4.10, 4.11 and 4.12.

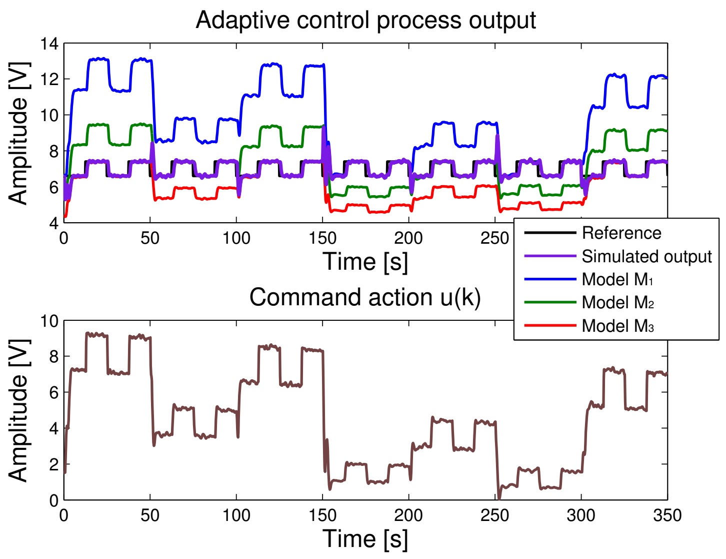


Figure 4.3: Adaptive control simulation with the autonomous system, with the three models available.

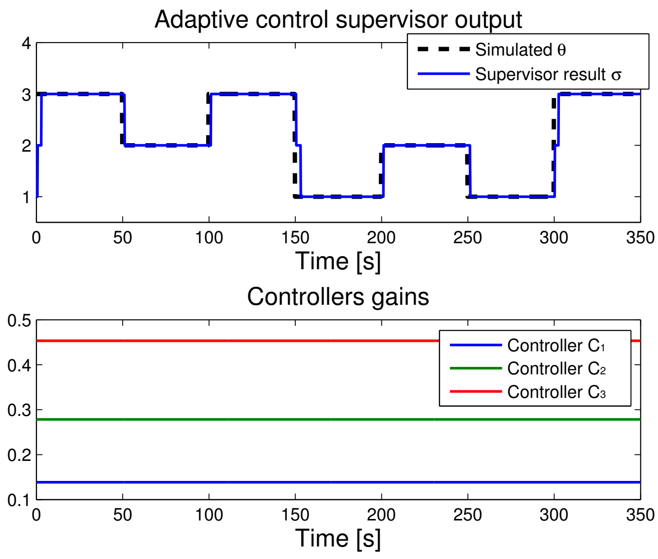


Figure 4.4: Supervisor index and controller gains for simulation of figure 4.3.

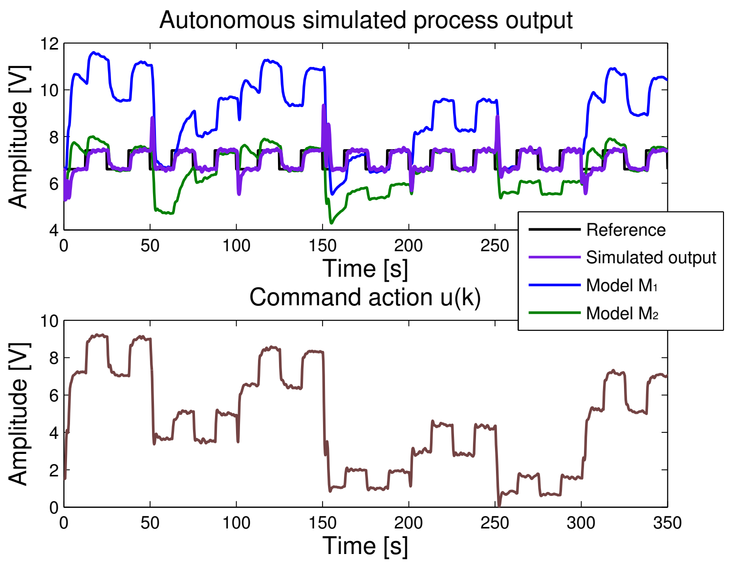


Figure 4.5: Nonadaptive simulation of the system with models M_1 and M_2.

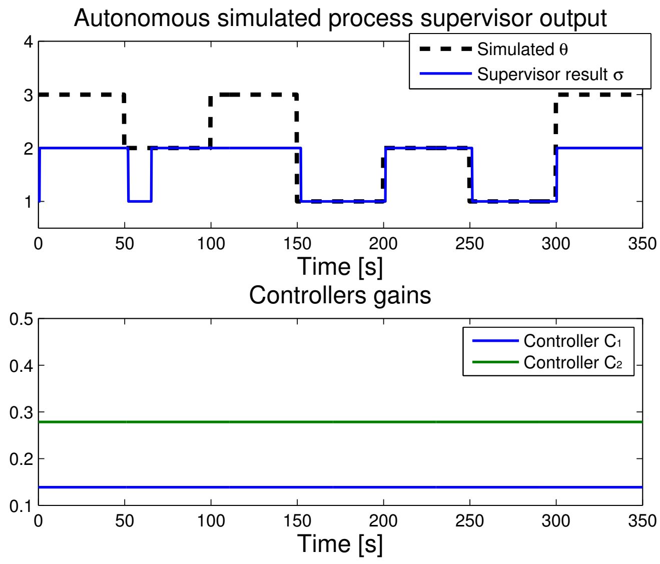


Figure 4.6: Supervisor index and controller gains for simulation of figure 4.5.

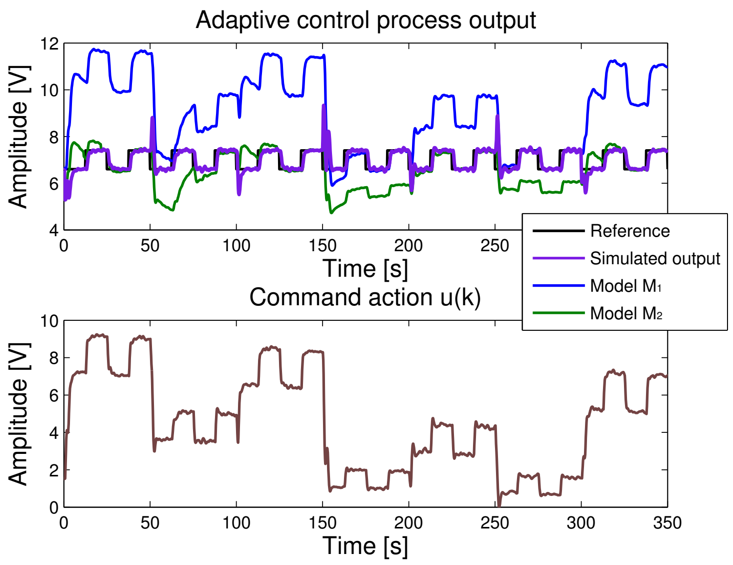


Figure 4.7: Adaptive control simulation of the system with models M_1 and M_2.

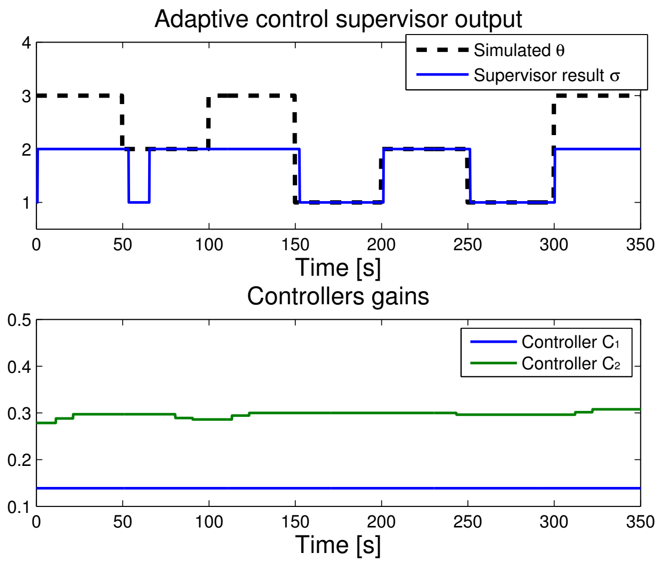


Figure 4.8: Supervisor index and controller gains for simulation of figure 4.7.

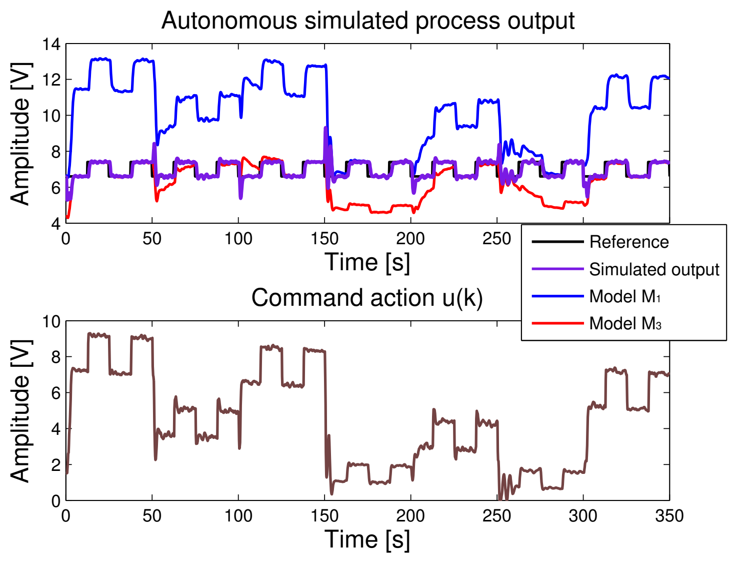


Figure 4.9: Nonadaptive simulation of the system with models M_1 and M_3.

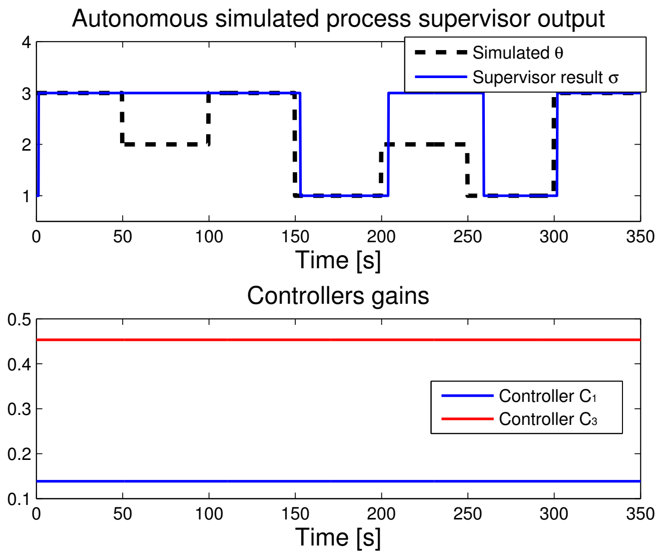


Figure 4.10: Supervisor index and controller gains for simulation of figure 4.9.

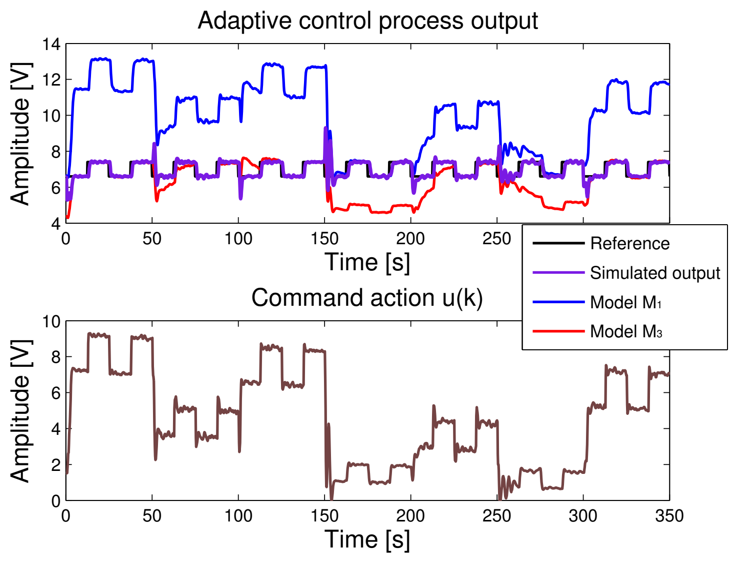


Figure 4.11: Adaptive control simulation of the system with models M_1 and M_3.

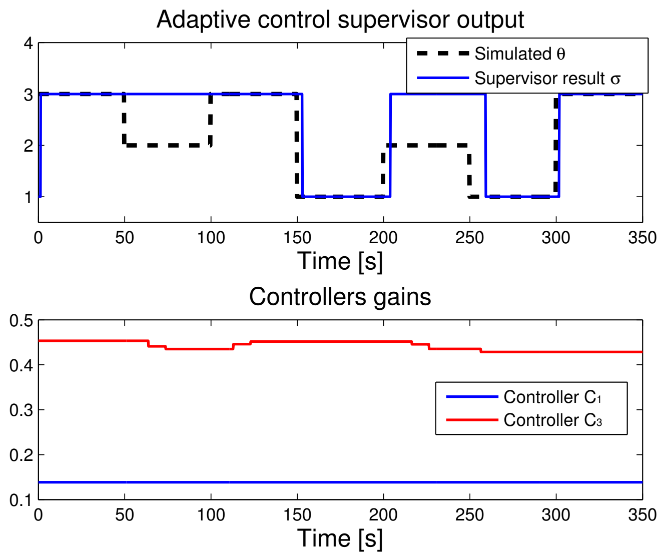


Figure 4.12: Supervisor index and controller gains for simulation of figure 4.11.

Although the last eight figures seem similar, we are able to see that, in the output performance of the adaptive control simulations, the estimated models M_j have a slightly best fit than in the nonadaptive simulations. As in figure 4.1, in this simulations the estimated models also converge to the 'real' one, which has the benefit of having an updated controller that improves the system performance.

In most part of the simulations, we can observe that the supervisor results are adequate. When the simulated throttle opening θ is higher than 1, the chosen models are M_2 in figures 4.6 and 4.8, and M_3 in figures 4.10 and 4.12. The model M_1 is also correctly identified.

The difference between the controllers gains, for nonadaptive and adaptive control, can also be viewed between figures 4.6 and 4.8, and between figures 4.10 and 4.12.

Moreover, we can observe that the changing controllers gains heads towards the controller gain correspondent to the current model, *i.e.*, C_2 heads to C_3 when $\sigma = 3$. However, the number of changes is limited, due to the thresholds predefined in table 4.1, which means that the controller gains does not change when δ_ν goes out of the bounds limited by δ_l and δ_u. The thresholds should be redefined when using a different number of models.

The project about adaptive control complements the autonomous performance of the system, where the output results are improved by identifying the current system model and update the supervisor and controller.

Chapter 5

Conclusion

In this work, we identified a nonlinear air heating system through multiple local models. Then, we designed a controller able to control the global system, with the support of a supervisor that can indicate which is the current model.

We conclude that the use of multiple controllers for reference tracking has an acceptable performance, and can be implemented on the air heating system.

Since the solution proposed in this work is not unique, we cannot state that using multiple controllers is the best way to deal with nonlinear systems. However, we can affirm that, for this specific process, the use of multiple LQG controllers has a better performance than the use of only one of them, to control the global system.

Due to the quick detection of the supervisor, the global system is able to regulate the output even when the throttle opening is still moving. That can be an important factor if the system stability depends on the quick controller adaptation.

We managed to overcome the overheating problem mentioned before with the offset estimation, which is also an important element for the supervisor quickness.

The implementation of adaptive control to the global system proved an improvement on the system performance. The adaptive control is important when the estimated models do not match the predicted ones due to small errors in the dynamic behavior. However, it becomes even more relevant when the number of models is insufficient and does not cover the entire dynamic range of the global system.

Moreover, the room temperature can influence the identification results, leading to different system measurements every day. Nevertheless, since the room temperature does not change the bandwidth of the system (section 2.1.2), the parametric identification can be repeated again at anytime, in order to obtain new models for the current conditions.

In the future, we may take into account more models, not only for more flows, but for different temperatures too. If more models are used, the supervisor may decrease its success rate. Nevertheless, if the supervisor chooses a model near to the correct one for control, the difference would be less significant, as the models would have similar dynamics.

Moreover, the supervisor logic can be improved to eliminate errors as initial conditions and when the controller saturates. In addition, another methods such as RST or PID controller can be also used. The results could then be compared with the achieved multiple LQG controllers performance.

Another aspects pointed in this work, like the variance of the room temperature and the overheating of the equipment itself, can be taken in account or externally controlled. The system offset effect may be modeled as well, instead of having an estimated offset, to complete the construction of the global model.

Bibliography

[1] R. Pickhardt, "Adaptive control of a solar power plant using a multi-model, *Control Theory and Applications*, vol. 147, n. 5, pp. 493-500, 2000.

[2] P. Oliveira, *et al.*, Supervised Multi-model Adaptive Control of Neuromuscular Blockade with Off-set Compensation, MSc. Dissertation, IST/UTL, Lisbon, Portugal, 2010.

[3] G. K. Thampi, et al., Multiple model based flight control design, in *Circuits and Systems*, 2002, pp. III-133-III-136 vol.3.

[4] L. Ljung, *System Identification - Theory For the User*, 2nd ed. Upper Saddle River, N.J.: PTR Prentice Hall, 1999.

[5] K. Ogata, *Modern Control Engineering*, 4th ed.: Prentice Hall, 2002.

[6] K. J. Åström and B. Wittenmark, *Computer-controlled Systems: Theory and Design*, 3rd ed.: Prentice Hall, 1990.

[7] G. F. Franklin, *et al.*, *Digital Control of Dynamic Systems*, 3rd ed.: Ellis-Kagle Press, 1997.

[8] R. Booton Jr, The measurement and representation of nonlinear systems, *Circuit Theory*, vol. 1, n. 4, pp. 32-34, 1954.

[9] D. Karlsson and D. J. Hill, Modelling and identification of nonlinear dynamic loads in power systems, *Power Systems*, vol. 9, n. 1, pp. 157-166, 1994

[10] W. J. Rugh, *Nonlinear system theory*, The Johns Hopkins University Press, 1981.

[11] T. A. Johansen and R. Murray-Smith, The operating regime approach to nonlinear modelling and control, in *Multiple Model Approaches to Modelling and Control*, R. Murray-Smith and T. A. Johansen, Ed., London, UK: Taylor and Francis, 1997, pp. 3-72.

[12] T. A. Johansen, et al., A software environment for gain scheduled controller design, *Control Systems Magazine, IEEE*, vol. 18, n. 2, pp. 48-60, 1998.

[13] J. M. Böling, *et al.*, Multi-Model Adaptive Control of a Simulated pH Neutralization Process, *Control engineering practice*, vol. 15, pp. 663-672, 2007.

[14] M. F. Rahmat, et al., Modelling Of PT326 Hot Air Blower Trainer Kit Using PRBS Signal And Cross-Correlation Technique, *Jurnal Teknologi D (42D)*, pp. 9-22, 2005.

[15] Manual of *Process Trainer PT 326*, Feedback Instruments Ltd.

[16] S. Weerasooriya and D. T. Phan, Discrete-time LQG/LTR design and modeling of a disk drive actuator tracking servo system, *Industrial Electronics*, vol. 42, n. 3, pp. 240-247, 1995

[17] I. R. Petersen, Guaranteed cost LQG control of uncertain linear systems, *Control Theory and Applications*, vol. 142, n. 2, pp. 95-102, 1995.

[18] J. Mošna and P. Pešek, "LQG tracking problem and its tracking robustness," The 11th Mediterranean Conference on Control and Automation, pp. 1-5, Mediterranean Control Association, Rhodes, Greece, 2003.

[19] Kumpati S. Narendra and O. A. Driollet, Adaptive control using multiple models, switching, and tuning, *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pp. 159-164, 2000.

[20] S. Haykin, *Adaptive Filter Theory*, 3rd ed.: Prentice Hall, 2002.

[21] L. Palma, et al., Adaptive Observer Based Fault Diagnosis Approach Applied to a Thermal Plant, *Proceedings of the 10th Mediterranean Conference on Control and Automation*, 2002.

[22] J. M. Lemos, *et al.*, Integrating Predictive and Switching Control: Basic Concepts and an Experimental Case Study, in *Nonlinear Model Predictive Control, Progress in Systems and Control Theory*, vol. 26, Part I, F. Allgöwer and A. Zheng, Eds.: Birkäuser, 2000, pp. 181-190.

[23] E. D. Gates, *Introduction to electronics*, 4th ed.: Delmar Thomson (Cengage) Learning, 2000

[24] G. Vinnicombe, "Frequency domain uncertainty and the graph topology," IEEE Trans. on Automatic Control, vol. 38, n. 9, pp. 1371-1383, 1993.

Appendix A

Heat energy exchange

The actuator of part ① in figure 2.1 can be presented as a scheme of energy exchanges, as shows in figure A.1.

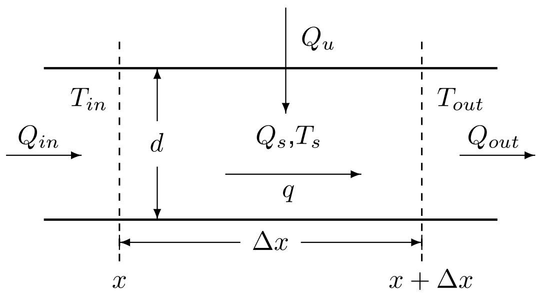


Figure A.1: Actuator scheme.

An energy balance yields that

$$Q_s = Q_{in} + Q_P - Q_{out} \quad , \tag{A.1}$$

where Q is the amount of heat transferred given by

$$Q = c_p m \Delta T \quad , \tag{A.2}$$

where c_p is the specific heat capacity, m is the air mass (in kg) and ΔT is temperature variation.

For the amount of heat stored, Q_s, we can have $c_p m = c_p \rho V = CV$, where ρ is the air density, $C = c_p \rho$ is the heat capacity and $V = A\Delta x$ is the volume, with $A = \frac{\pi d^2}{4}$ the section of the tube. The incremental temperature variation is given by $\Delta T = T_s(t+\Delta t) - T_s(t)$, thereby we have

$$Q_s(t) = CV\big[T_s(t+\Delta t) - T_s(t)\big] \quad , \tag{A.3}$$

where the stored temperature, T_s, can be the temperature anywhere between x and $x+\Delta x$.

For the amount of heat that enters and exists the actuator, we do not have a volume V. Instead, we make $c_p m = CV = Cq\Delta t$, where q is the airflow that passes in the tube in the time of Δt. Therefore, we have

$$Q_{in}(t) = Cq\,\Delta t\,T_{in}(t) \quad , \quad Q_{out}(t) = Cq\,\Delta t\,T_{out}(t) \quad , \tag{A.4}$$

where the inflowing temperature T_{in} is located in x and the outgoing temperature T_{out} is located in $x+\Delta x$.

We can achieve equation (2.1) with making the temperatures as

$$T_{in}(t) = T(x,t) \quad , \tag{A.5}$$

$$T_{out}(t) = T(x+\Delta x, t) \quad , \tag{A.6}$$

$$T_s(t) = T_{in}(t) = T(x,t) \quad , \tag{A.7}$$

$$T_s(t+\Delta t) = T(x, t+\Delta t) \quad , \tag{A.8}$$

although, instead of equation (A.7) and (A.8), we could have the stored temperature as

$$T_s(t) = T_{out}(t) = T(x+\Delta x, t) \quad , \tag{A.9}$$

$$T_s(t+\Delta t) = T(x+\Delta, t+\Delta t) \quad , \tag{A.10}$$

The amount of heat inputted to the actuator is given by

$$Q_u(t) = P(t)\Delta t \tag{A.11}$$

which means that Q_u is the power of the heat injected in the system P over time.

Appendix B

State space model matrices

The state space matrices for each model are

$$A_1 = \begin{bmatrix} 1.1310 & -0.3862 & 0.0596 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} , \quad C_1 = \begin{bmatrix} 0 & 0.0562 & 0.1103 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 1.0264 & -0.3446 & 0.0576 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} , \quad C_2 = \begin{bmatrix} 0.0148 & 0.1268 & 0 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 0.8513 & -0.1953 & 0.0247 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} , \quad C_3 = \begin{bmatrix} 0.0276 & 0.0981 & 0 \end{bmatrix}.$$

Appendix C

Controller and observer gains

The LQG gains for each model, computed through equation (3.6), are

$$K_1 = \begin{bmatrix} 0.0190 & -0.0081 & 0.0016 & 0.2644 \end{bmatrix},$$

$$K_2 = \begin{bmatrix} 0.0549 & -0.0228 & 0.0049 & 0.5037 \end{bmatrix},$$

$$K_3 = \begin{bmatrix} 0.3621 & -0.0938 & 0.0138 & 1.3950 \end{bmatrix},$$

$$L_1 = \begin{bmatrix} 3.3492 & 3.0646 & 2.6626 & 0.7847 \end{bmatrix}^T,$$

$$L_2 = \begin{bmatrix} 2.8008 & 2.5676 & 2.2048 & 0.8261 \end{bmatrix}^T,$$

$$L_3 = \begin{bmatrix} 2.4510 & 2.2824 & 2.0094 & 0.8601 \end{bmatrix}^T.$$

Appendix D

First order low-pass filter

A discrete low-pass filter follows

$$out(k) = \lambda \, out(k-1) + (1-\lambda) \, in(k) \quad ,$$

with input $in(k)$, output $out(k)$ and λ is the pole of the discrete transfer function

$$out(k) = \frac{(1-\lambda)q}{q-\lambda} \, in(k) \quad .$$

Appendix E

Variable forgetting factor

The variable forgetting factor follows the next block diagram.

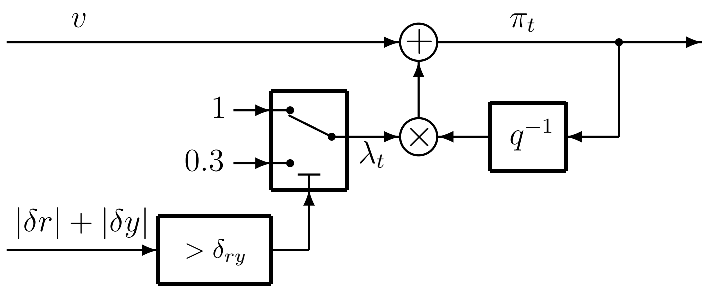


Figure E.1: Block diagram for the variable forgetting factor implementation.

Appendix F

Offset estimation with forgetting factor

Based on [2], the system model is defined by

$$Az = Bu \quad , \quad y = z + b \quad , \tag{F.1}$$

For these models, the estimated output can be

$$\hat{y}(k) = \left(1 - \frac{A(q)}{\omega(q)}\right) y(k) + \frac{B(q)}{\omega(q)} u(k) + \frac{A(q)}{\omega(q)} b(k) \quad . \tag{F.2}$$

The output estimation error is given by[1]

$$\hat{e}(k) = \hat{y}(k) - y(k) = \alpha(k) - b(k)\beta(k) \quad , \tag{F.3}$$

where

$$\alpha(k) = -\frac{A(q)}{\omega(q)} y(k) + \frac{B(q)}{\omega(q)} u(k) \ , \ \beta(k) = -\frac{A(q)}{\omega(q)} u_{-1}(k) \quad . \tag{F.4}$$

The cost function to be minimized, including the forgetting factor λ_b, is[2]

$$J\big(b(k)\big) = \sum_{j=0}^{k} \lambda_b^{k-j} \Big(\hat{y}(k) - y(k)\Big)^2 = \sum_{j=0}^{k} \lambda_b^{k-j} \Big(\alpha(j) - b(k)\beta(j)\Big)^2 \quad , \tag{F.5}$$

[1] Incorrectly in [2] is written $\alpha(k) + b(k)\beta(k)$. This would be valid only if $\beta(k) = +\frac{A(q)}{\omega(q)} u_{-1}(k)$.

[2] Previous values of $b(k)$ are not used in the cost function. That is why we do not use $b(j)$.

that is equivalent to

$$J\big(b(k)\big) = \sum_{j=0}^{k} \lambda_b^{k-j}\alpha^2(j) - 2b(k)\sum_{j=0}^{k}\lambda_b^{k-j}\alpha(j)\beta(j) + b^2(k)\sum_{j=0}^{k}\lambda_b^{k-j}\beta^2(j)\,. \quad \text{(F.6)}$$

The partial derivative of $J\big(b(k)\big)$ of equation (F.6) with respect to the variable $b(k)$ is

$$\frac{\partial J\big(b(k)\big)}{\partial b(k)} = -2\sum_{j=0}^{k}\lambda_b^{k-j}\alpha(j)\beta(j) + 2b(k)\sum_{j=0}^{k}\lambda_b^{k-j}\beta^2(j) \quad . \qquad \text{(F.7)}$$

To have the estimation of $b(k)$, we make equation (F.7) equal to zero

$$\left.\frac{\partial J\big(b(k)\big)}{\partial b(k)}\right|_{b(k)=\hat{b}(k)} = 0 \qquad \Rightarrow \qquad \hat{b}(k) = \frac{\sum_{j=0}^{k}\lambda_b^{k-j}\alpha(j)\beta(j)}{\sum_{j=0}^{k}\lambda_b^{k-j}\beta^2(j)} \quad . \qquad \text{(F.8)}$$