



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



departamento de ingeniería
de sistemas y automática

Master's Degree in Automatic and Industrial Computing

Modeling, identification and control of an elastic joint

Author:

Rafael Sergio Celis García

Teacher:

Ranko Zotovic Stanisic

Convocatoria: Septiembre 2018

ABSTRACT

Nowadays the use of harmonic drive systems has become common because its high reduction ratio and high torque transmission in a compact mechanism, especially in robotics where the weight of the robot it's an important factor, lower weight transmission with high torque means less energy is spent for operation.

On the other hand, the harmonic drive transmissions have more complex dynamic behaviour than conventional gear transmissions. In this project, an experimental platform has been set up in order to analyse the behaviour of the polymeric low cost Harmonic Drive transmission that will be integrated into the joints of the WRAD robot, a prototype developed in previous projects to assist disabled persons. The first part of the project is focused on the experimental platform materials used as the BLDC Motor, the ESCON power stage and the Discovery board mainly. Its configuration as well as the connections and building in order to acquire data from the real system are explained.

After that, the different mathematical models that have been researched in the last years are studied in order to integrate them in a Grey-box model to identify the parameters.

Finally, the results are tested under simulation and results between the simulations and the real data are compared as well as the difference between models. The mathematical modelling of the Harmonic Drive transmission is reviewed, from the most simple to the most complex model containing nonlinear friction, hysteresis and non-linear cubic elasticity, this will allow in the final part of the project to compare the results obtained and integrated in those models.

As long as the project is developed in a low cost prototype platform some limitations were found while working with the system in order to achieve the goals proposed, this limitations would have a repercussion on later results, anyway most of them are solved or treated to reduce its impact in the project.

All the process involved in the data acquisition with the Discovery STM32F4 board will be explained together with the signal processing applied before the identification procedure where different models are identified, from the most basic to the most complex with non-linearities, which allows comparing the accuracy of the different models. In order to simplify the identification task, some parameters are estimated from different experiments designed to appreciate the physical phenomena in the data taken. This will be useful in order to reduce the number of parameters being identified, which might lead to bad model identification due to local minima. In addition, the different methods for prediction error minimization used by the matlab functions are explained, and then the model is estimated in steps according to different data sets of the same system with different inputs. The parameters obtained in the identification will be simulated with the models studied and according to the results obtained the parameter variation will allow us to study how it affects the response of the system, this will be important for later conclusions. In addition, the models that have not been identified will be simulated, as well as the effect of the variation of the parameters in the model.

Key words: BLDC motor, Harmonic Drive, Elastic robot joint, Identification, Grey-box modelling, Matlab IDNLGREY, Discovery STM32F4.

RESUMEN

Hoy en día el uso de reductores del tipo Harmonic Drive se ha vuelto algo común debido a sus propiedades ideales para robótica como gran ratio de reducción y gran transmisión de par en un mecanismo compacto, esto es imprescindible en robótica donde el peso juega un papel muy importante, especialmente en lo que a ahorro de energía se refiere durante el funcionamiento del robot. Por otra parte, este tipo de transmisiones suponen un reto a la hora de ser modeladas y controladas ya que presentan un comportamiento dinámico mucho más complejo que las transmisiones convencionales. Este proyecto se ha llevado a cabo en una plataforma experimental montada en el departamento DISA, donde se dispone de un prototipo diseñado previamente por otros estudiantes dirigidos por el profesor Ranko Zotovic, se trata del brazo robot para asistencia a discapacitados "WRAD", en él se han empleado reductores Harmonic Drive de bajo coste de material polimérico como transmisión para los motores de los eslabones 1 y 2.

La primera parte del proyecto se centra en explicar la plataforma experimental, así como el material del que se dispone, motores BLDC, etapa de potencia ESCON, tarjeta Discovery, encoders y todas las conexiones y montaje correspondiente para la toma de datos del sistema real con el que realizar en pasos posteriores la identificación. Al tratarse de una plataforma de bajo coste se han encontrado algunas limitaciones durante el desarrollo del proyecto, sin embargo en la medida de lo posible se ha tratado de solventar estos problemas con el material del que se dispone.

En siguientes capítulos los diferentes modelos de Harmonic Drive propuestos por investigadores expertos en la materia que han sido estudiados y analizados con equipamiento profesional son presentados para su posterior implementación en Matlab en lo que se conoce como modelo de caja gris o "Grey-box modeling" donde se obtiene un modelo parametrizado donde el objetivo es encontrar el valor de dichos parámetros que proporciona una respuesta lo más parecida posible a los datos medidos del sistema.

Para la identificación con los datos obtenidos con la tarjeta Discovery, se explica el proceso a seguir paso a paso para la obtención del vector que contiene dichos datos. Estos datos han de ser preprocesados antes de identificar y se evaluará el efecto de dicho pre procesamiento al comparar modelos obtenidos aplicando dicho paso y sin aplicarlo.

Finalmente los resultados son evaluados por medio de diferentes indicadores que permiten determinar qué grado de exactitud tiene el modelo obtenido, también se compararán las respuestas de diferentes modelos y se verá cómo la inclusión de fenómenos físicos adicionales puede proporcionar una mejora del modelo obtenido. Debido a la complejidad del sistema, se dividirá la identificación en varios pasos tratando de aislar lo máximo posible el valor de los parámetros en base a la información que se dispone del sistema, esto ayudará a que el proceso de optimización de la identificación encuentre mínimos locales, lo que conlleva a un modelo incorrecto.

Por último se presenta un generador de trayectoria de cuarto orden junto a simulaciones de los modelos obtenidos variando los parámetros característicos de la transmisión.

Palabras clave: BLDC motor, Harmonic Drive, Elastic robot joint, Identification, Grey-box modelling, Matlab IDNLGREY, Discovery STM32F4.

GENERAL INDEX OF DOCUMENTS

- DOCUMENT I: PROJECT MEMORY
- DOCUMENT II: ANNEXES

DOCUMENT I: PROJECT MEMORY

1. Introduction.....	1
2. System setup	4
2.1 Mechanical structure.....	4
2.1.1 Prototype robot structure	4
2.1.2 Harmonic drive	5
2.2 Electric and electronic components.....	7
2.2.1 BLDC motor	7
2.2.2 Encoder headers and codewheels.....	8
2.2.3 Discovery STM32F429Zi	11
2.2.4 EPOS/ESCON modules	12
2.2.6 Auxiliary electronic circuits	26
2.2.7 System architecture	27
3. Modelling robots with flexible joints.....	29
3.1 Modelling the Harmonic Drive transmission in a robot joint.....	30
3.2 Harmonic Drive state space modelling.....	34
3.3 Modelling non-linearities	35
3.3.1 Hysteresis spring	35
3.3.2 Non-linear friction modelling	38
3.3.3 Backlash	41
4. System limitations	43
4.3.1 Encoder saturation	43
4.3.2 Velocity estimation.....	45
4.3.3 DAC mismatch	47
4.3.4 Speed saturation in current control	49

4.3.5 ESCON data recorder low sampling time	52
5. Identification	54
5.1 Methods for nonlinear least squares curve-fitting problems	59
5.1.1 The gradient descent method	60
5.1.2 Gauss-Newton method	61
5.1.3 The Levenberg-Marquardt method.....	62
5.2 Reduction ratio comprobation	62
5.2.1 Constant step	63
5.2.2 Stairs.....	64
5.2.3 Sinusoidal	65
5.3 Voltage control identification.....	66
5.3.1 Static friction parameters.....	66
5.3.2 Rigid body parameters	70
5.3.3 Harmonic drive (simplified model) parameter identification	77
5.3.4 Harmonic drive with nonlinear friction and elasticity parameter identification	93
5.4 Current control identification.....	100
5.4.1 Harmonic drive with nonlinear friction and elasticity parameter identification	100
5.5 Frequency response analysis.....	107
6. Simulations and trajectory generator	110
6.1 Simulations with the parameters identified.....	110
6.1.1 Model simulation	110
6.1.2 Bouc-Wen model.....	113
6.2 Fourth order trajectory generator for Keil uVision	116
7. Conclusions and future works	118
8. Budget	119
9. References.....	121

FIGURES'S INDEX

Figure 1. Harmonic Drive transmission components.....	2
Figure 2. Built prototype CAD.....	4
Figure 3. Robot arm built systematically.....	5
Figure 4. Harmonic Drive used for the project.....	5
Figure 5. Harmonic Drive specified dimensions.....	6
Figure 6. Correct assembly.....	6
Figure 7. EC motor dimensions.....	7
Figure 8. EC motor used for the project.....	7
Figure 9. Encoder and codewheel HEDS-9100.....	9
Figure 10. Encoder channels signal.....	9
Figure 11. Encoder header dimensions.....	10
Figure 12. BLDC Motor with encoder header and codewheel mounted.....	10
Figure 13. Discovery STM32F429Zi.....	11
Figure 14. ESCON module 50/5.....	13
Figure 15. ESCON module 50/5 dimensions and pin assignment.....	14
Figure 16. Enable pin configuration in ESCON Studio.....	15
Figure 17. Motor type select.....	15
Figure 18. Motor parameters from datasheet.....	16
Figure 19. Motor operating point specified in datasheet.....	16
Figure 20. Hall sensor pattern configuration.....	17
Figure 21. Current control ESCON diagram.....	17
Figure 22. DAC input, Current averaged and motor speed in the ESCON data recorder.....	18
Figure 23. Set point configuration for current control.....	18
Figure 24. Offset needed for operating in two directions.....	19
Figure 25. Current control ESCON monitored.....	19
Figure 26. Proportional current controller tuning.....	20
Figure 27. Voltage control diagram.....	20
Figure 28. Power stage gain definition.....	21
Figure 29. Current limitation and speed control ramp configuration.....	21

Figure 30. Speed control offset.	22
Figure 31. ESCON studio speed control (open loop).....	22
Figure 32. ESCON power supply cable.....	23
Figure 33. ESCON motor cable.	24
Figure 34. Hall sensor cable.....	25
Figure 35. ESCON Module and the motor cables.....	25
Figure 36. System connections diagram.	26
Figure 37. Electronic circuit in the power stage.....	27
Figure 38. Experimental platform mounted in the DISA.....	28
Figure 39. Elastic joint approach.	29
Figure 40. Harmonic Drive configuration used in the application.	31
Figure 41. Current control block diagram.	33
Figure 42 Voltage control block diagram	33
Figure 43. Nonlinear elasticity (cubic polynomial) Simulink implementation.	36
Figure 44. Non-linear elastic polynomial.....	36
Figure 45. Typical hysteresis shape in Harmonic Drives.	37
Figure 46. Internal state of the hysteretic differential equation.	38
Figure 47. LuGre Simulink implementation.....	39
Figure 48. Striebeck function Simulink implementation.....	39
Figure 49. General view of the elastic joint implemented in Simulink.	40
Figure 50. One joint with non-linearities Simulink representation.....	40
Figure 51. Backlash in the gear transmission contact teeth.	41
Figure 52. Gear transmission with elasticities and backlash.	42
Figure 53. Encoder readings with timer auto reload.	43
Figure 54. Encoder position fixed.....	44
Figure 55. "Pulses" from zero to 3.14 estimated when moving at low speeds.	45
Figure 56. Quantization error in the encoder two measurements.....	46
Figure 57. Velocity 2 filtered with no phase filter "filtfilt".....	46
Figure 58. Velocity 2 after medfilt with values 1, 10 and 50.....	47
Figure 59. ESCON analog input2.....	48
Figure 60. DISCOVERY input voltage data.	48

Figure 61. Input with initial peak for friction break.	50
Figure 62. Input signal, step at 0.9V.	50
Figure 63. ESCON data record showing speed (voltage) saturation and current (torque) saturation when reached the maximum speed.	51
Figure 64. ESCON Studio data recorder.	52
Figure 65. Current acquired from ESCON and resampled signal.	53
Figure 66. Terminal receiving data example.	54
Figure 67. Flux diagram of the identification procedure.	55
Figure 68. ESCON studio data recorder configuration.	56
Figure 69. Compile and load buttons on Keil uVision.	56
Figure 70. Code writing data on the SDRAM.	56
Figure 71. Data is not transmitted until the state is true.	57
Figure 72. Teraterm connection options.	57
Figure 73. Log save screen.	57
Figure 74. Matlab data script.	58
Figure 75. Velocity filter applied.	58
Figure 76. Kinematic error estimation.	59
Figure 77. Signal plot script.	59
Figure 78. Constant input voltage.	63
Figure 79. Motor and link positions for constant input voltage.	63
Figure 80. Stairs input voltage.	64
Figure 81. Motor and link position for stair input voltage.	64
Figure 82. Sum of sinusoidals input voltage.	65
Figure 83. Encoder 1 and 2 measurements for the given experiment.	65
Figure 84. Input step and velocity response.	66
Figure 85. Steady state velocity vs. DAC Voltage input.	66
Figure 86. Velocity plot of the data set 1.	71
Figure 87. Simulated response vs. real data.	74
Figure 88. One-step prediction error.	74
Figure 89. Amplitude of one of the "bars" equal to pi.	75
Figure 90. Residuals of the model estimated.	75

Figure 91. Input signal plot for data set "sins1"	78
Figure 92. Input signal plot for data set "scs1"	79
Figure 93. Input signal plot for data set "scs2"	79
Figure 94. Input signal plot for data set "bdsins"	80
Figure 95. Input signals of the data sets.....	84
Figure 96. Simulated response vs. real data measurements.	86
Figure 97. Residuals of the obtained model.....	87
Figure 98. Noise velocity2 estimated response.	89
Figure 99. Filtfilt velocity2 estimated response.	89
Figure 100. Residual correlation analysis for the identified models.....	92
Figure 101. Left side residuals from first ident, Right side residuals from last ident.....	92
Figure 102. Identified NLGR with one direction simulated response compared with both directions data set.....	93
Figure 103. Inputs used for both directions movement and friction identification.	94
Figure 104. Simulated response comparison for the model obtained and the validation data set.	97
Figure 105. 1-Step prediction error for the identified model.	98
Figure 106. Residue correlation for the estimated model.	98
Figure 107. Step response for the identified model.	99
Figure 108. Current response for high frequency DAC input.....	100
Figure 109. Low frequency input for current control identification.	100
Figure 110. Simulated response comparison for the first data set.....	103
Figure 111. Simulated response comparison for the second data set.....	104
Figure 112. 1-Step prediction error.....	104
Figure 113. Residue correlation.	105
Figure 114. Step response of the identified system.....	106
Figure 115. Typical bode plot shape of link velocity respect to the current(left), and motor velocity respect to the current (right)	109
Figure 116. General view of the elastic joint model implemented in Simulink.	110
Figure 117. Elastic joint Simulink implementation.....	111
Figure 118. Trigonometric function for the friction implemented on Simulink.....	111
Figure 119. Simulink real data vs simulated signal.....	112
Figure 120. Bouc-Wen model implementation in simulink.....	113

Figure 121. Bouc-Wen subsystem for plastic-elastic balance 113

Figure 122. Bouc-Wen internal state Simulink implementation 114

Figure 123. Hysteresis shape obtained by simulation 115

Figure 124. Friction force vs velocity, x axis velocity, y axis friction force 115

TABLES INDEX

Table 1. Harmonic Drive manufacturer specifications.....	6
Table 2. EC motor parameters.	8
Table 3. Discovery board pins used.....	12
Table 4. Electrical rating properties.	13
Table 5. Inputs and outputs used.....	13
Table 6. Pin assignment table.....	14
Table 7. ESCON power cable information.	23
Table 8. ESCON motor cable information.	24
Table 9. Hall sensor cable information.....	24
Table 10. Motor response in current control mode.	50
Table 11. Motor response in current control mode.....	51
Table 12. Motor response in current control mode.....	51
Table 13. Estimated reduction ratio from the three experiments.....	65
Table 14. Friction parameters identified.....	69
Table 15. Parameters identified in the first try.	73
Table 16. Parameters identified in the second try.....	73
Table 17. Parameters used in the next step.....	73
Table 18. Data set properties.	77
Table 19. Data set properties.	78
Table 20. Data set properties.	79
Table 21. Data set properties.	80
Table 23. Identification 1 procedure info.....	84
Table 24. Initial parameters values and boundaries.	85
Table 25. Final estimated parameters for both velocity 2 filter data sets.	85
Table 26. Final parameter value after transformation.....	86
Table 27. Identification 2 procedure info.....	87
Table 28. Identification 2 initial parameters and its boundaries.	88
Table 29. Final estimated parameters for both velocity 2 filter data sets.	88
Table 30. Final values obtained after transformation.....	89

Table 30.Results comparison between identification 1 and 2. 90

Table 31.Identification friction initial parameters and its boundaries. 95

Table 32. Final estimated parameters for both velocity 2 filter data sets. 95

Table 33.Identification friction initial parameters and its boundaries. 101

Table 34. Final estimated parameters for both velocity 2 filter data sets. 101

Table 35.Material budget. 119

DOCUMENT I:

PROJECT MEMORY

Notation and abbreviations

Notation	Meaning
\dot{x}	First derivative
\ddot{x}	Second derivative
\hat{x}	Estimated
θ	Reference to motor variables
τ	Torque
ρ	Parameter vector
δ	Generalized coordinate vector (Multivariable case)
ϑ	Rayleigh function
$\Delta\theta$	Kinematic error
V_{supp}	Supply voltage
LS	Least squares
SDRAM	Synchronous dynamic random access memory
D.O.F	Degree(s) of freedom
LSQNONLIN	Least squares nonlinear search
GNA	Gauss Newton adaptive search
LM	Levenberg-Marquardt search
EMF	Electromotive force
H.D	Harmonic Drive
NLGR	Nonlinear grey box
DAC	Digital to analog converter
VCP	Virtual com port

1. Introduction

During the past decades, industrial robots have become a very important factor in the manufacturing industry. Nowadays robots can be found in new areas, because of this and to be able to go into new markets, the new robots require better performance or lower price. To achieve these goals the physical robot structures are built lighter and weaker, the unique performance features of harmonic drives, such as high gear ratios and high torque capacities in a compact geometry, justify their use as mentioned above.

Since the 60's the harmonic drive has found widespread acceptance, this mechanical transmission employs a continuous deflection wave along a non-rigid gear to allow for gradual engagement of gear teeth. Because of this unconventional gear-tooth meshing action, harmonic drives can deliver high reduction ratios in a very small package; this particular way of operation has created a new area of exploration and understanding. The harmonic drive exhibits performance features both superior and inferior compared to conventional gear transmissions. Its performance advantages are:

- High torque capacity.
- Concentric geometry.
- Lightweight and compact design.
- Near-zero backlash.
- High efficiency and back drivability.

Harmonic drive systems however have some disadvantages as:

- High flexibility.
- Resonance vibration.
- Friction and structural damping nonlinearities.

The unique performance features of the harmonic drive allows it to be used in many different fields, from space robots, assembly equipment to measuring instruments or like in this project for wearable robot arms for its lightweight and compact geometry specifically.

In the last years, there has been an increasing use of the technologies in the robotic field, as usual, new technologies are applied first at the industrial sector and after that, it is applied to another social fields, by means of this tools robots can be used to be exposed in dangerous zones to keep safe workers health.

In the industrial sector, last years with the industry 4.0 the collaborative robotics have seen its demand increased, this kind of robots allows to work together with the workers with its tasks without the need of barriers or security systems between the robot and the person.

To achieve the security of the user force and torque control that allows controlling and preventing collisions. This control system also allow an easier programming for the user.

However, harmonic drives can exhibit surprisingly more complex dynamic behaviour than conventional gear transmissions. As higher demand on the accuracy of the robot models used in the

Modelling, identification and control of an elastic joint.

controllers are growing, this means good models are needed for the new applications, so there is a need for an automated way of getting mathematical models and also the estimation of its parameters.

It has to be taken into account that in robot identification, the amount of sensors is limited; usually the only variable that is measured is the position of the motors. Others also include a torque sensor for estimating physical phenomena such as mechanical hysteresis and different friction models [4],[6], in this case with the test platform used there are two encoders in the motor and link side so both measurements are available, also some parameters are isolated for identifying some physical properties of the motor-gear transmission structure.

This project is part of a bigger project where the goal is to design a wearable robot arm prototype for assistance to people with reduced mobility in the superior body extremities. All the mechanic structure has been designed and developed previously in past projects; also, some simulations and control for the 2-link arm structure have been developed before.

The robotic arm is thought to be worn in a vest, and it will be supplied with batteries.



Figure 1. Harmonic Drive transmission components.

In the present project, the gear is built into a BLDC Motor from the Maxon motor company, which are characterized by high efficiency, wide rotational speeds range and long lifespan, the motor corresponds to the “flat” lines of the motors manufactured by Maxon. When modelling the motor in later chapters it will be considered as a DC motor for modelling purposes.

In robotics, the motors are actuated by means of the so-called power stage. The power stage selected for this project was the ESCON Module 50/5 from Maxon, which is specified to work with the motor available. The different modes of operation as well as the overlay of the ESCON Module will be described and the setup procedure will be explained.

For the data acquisition the Discovery STM32F4 board developed by STMicroelectronics is used recording the encoder measurements from both sides of the elastic joint, after storing the data in the

Discovery SDRAM, it is transmitted to the PC via USB by means of the VCP com port, this data will be used for the identification procedure.

This data has to be preprocessed in order to make a proper identification, in the section 4, when explaining the system limitations, the data processing will be explained.

Once the data has been treated, the identification procedure is design, depending on the physical behaviour that will be identified, different model approaches will be used, also the data will be informative enough about this physical phenomena in order to achieve a good identification.

The model is implemented in a MEX file that will contain the model state equations as well as the parameters of that precise model. Then in a Matlab script, the steps are the following:

- Define name and units for the input, states and outputs.
- Define the parameter vector with its boundaries.
- Define the initial states from the data, in this case as the outputs are directly the states, we can define the initial states precisely.
- Integrate the model defined in the MEX file into the Matlab script.
- Create an iddata object containing all the data sets, that will be called later on each step.
- Estimate the model until there is one data set left, this one will be used for validation purpose.
- Each estimation step saves the parameter values and use them as initial guess for the next estimation step.
- Compare the identified models in each estimation step with the validation data set.

When estimating the model, different criterion of fit or searchmethod can be used, these are explained in later sections.

2. System setup

In this chapter, the whole system used for the purpose of this project will be explained; this point has been divided into mechanical equipment and electric and electronic. During the project some things were changed, here the final version used will be described and some of those changes needed from the initial version of the system will be explained.

Jose Antonio Mulet Alberola and Cristóbal Haro Galarza, who were directed by the teacher Santiago Gutierrez, designed the robot structure in previous works. For the Discovery, other students that developed its projects with Ranko developed configuration and functions in previous works. Anyway when starting the project to understand the whole system and to make it work properly many things had to be checked and changed such as encoder headers that were burnt, codewheels damaged and the flexpline of the Harmonic drive dropped after operating with it during some period of time. All this problems had to be fixed in order to start the latter steps of the project; the mechanical problems with the structure were fixed by adding new pieces or fixing the structure treating it with chemical products.

All the mechanical materials as well as the electric and electronics are detailed in the following section.

2.1 Mechanical structure

2.1.1 Prototype robot structure

The robot structure was built in a 3D printer, with ABS as main material, after some years and taking into account previous works, the first challenge presented was that new pieces needed for the proper work of the system were needed, these are used for fixing the flexpline and the plastic ring used for the first motor.



Figure 2. Built prototype CAD.

Modelling, identification and control of an elastic joint.

As a main feature for this project, a new piece was designed in order to keep the flexpline inside the circular spline, because with the previous design, these two-pieces would separate during operation.



Figure 3. Robot arm built systematically.

2.1.2 Harmonic drive

Explain the harmonic drive use, its properties, its use in nowadays robots, possible configurations and the configuration chosen for our robot (CS fixed, WG input, FS output, with $-1/N$ reduction).



Figure 4. Harmonic Drive used for the project.

The main goal of this project consists in the identification of the Harmonic Drive developed by the Harmonic Drive polymer GMBH Company. This specific model was developed few years ago and it cannot be bought nowadays since it is not being produced anymore, also the specifications given by the manufacturer are very poor, this can be consulted in the annexes where all the datasheets and pdfs from the materials used for this project are presented.

Some of the technical specifications provided are shown in the following table.

Modelling, identification and control of an elastic joint.

Table 1. Harmonic Drive manufacturer specifications.

Reduction ratio	70
Rated torque (Nm)	6
Peak torque (N)	16
Efficiency (rated operation)	61%
Rated input speed (rpm)	3000
Maximum input speed (4000 rpm)	4000
No-load running torque (0.05 Nm)	0.05
Weight (g)	149
Repeatability (arcmin)	+/- 0.2

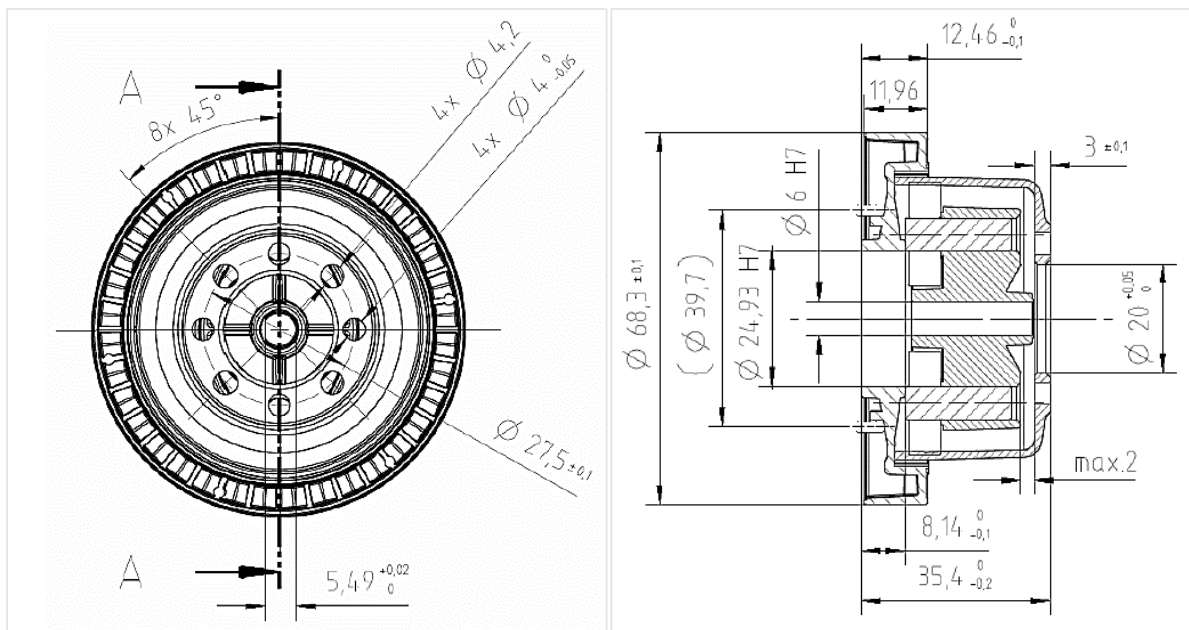


Figure 5. Harmonic Drive specified dimensions.

The gear is attached to an EC motor and built inside the prototype arm structure, when assembled properly the dedoidal form shown below should be avoided.

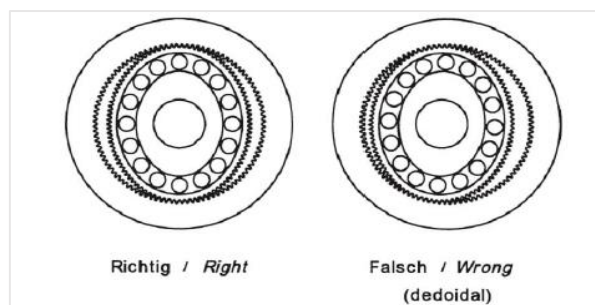


Figure 6. Correct assembly.

2.2 Electric and electronic components

2.2.1 BLDC motor

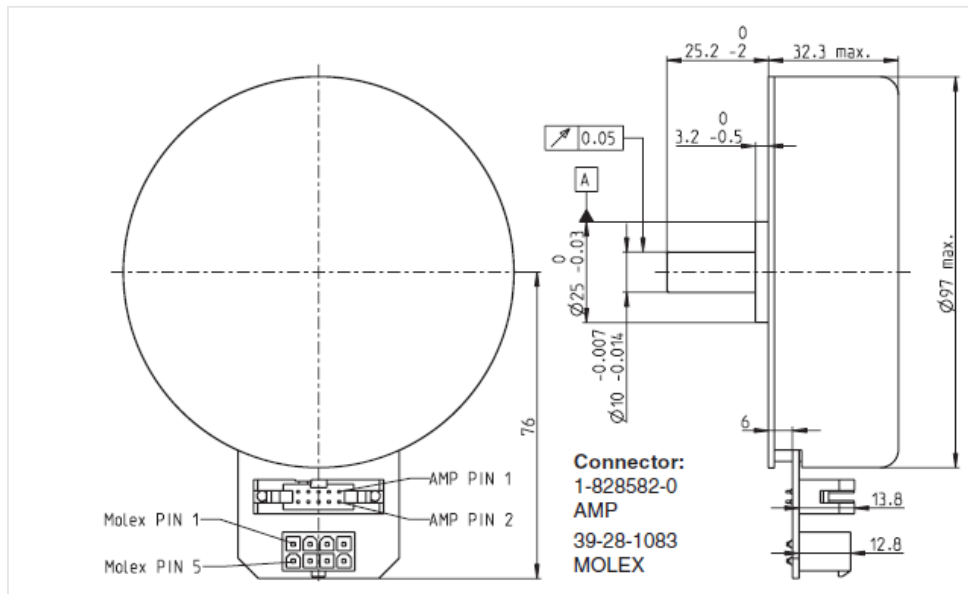


Figure 7. EC motor dimensions.

All electric motors work by the interaction of two magnetic fields pushing on each other. One field is created by the rotor and one by the stator. The difference between motor types is in how these fields are created and controlled. BLDC motors use permanent magnets to create the rotor field, and a series of coils controlled by an electronic controller or commutator to create the stator field.

The fact of being brushless avoids the sparking and short life of brushed motors, so they are rapidly becoming more popular than the brushed DC motors used conventionally as long as they need less maintenance and have longer life span. Because they have electronics controlling the stator this motors don't need to waste power inducing the rotor field, they give better performance and controllability, and run cooler than induction. In addition, they are very high efficiency, and maintain a high efficiency level at part speed.



Figure 8. EC motor used for the project.

Some of the parameters of the motor can be directly extracted from the datasheet, the most important are:

Table 2. EC motor parameters.

Nominal Voltage (V _{supp})	36.0
Nominal speed (rpm)	1240
Nominal current (A)	1.78
Nominal torque (mNm)	405
Terminal resistance phase to phase (Ohm)	2.3
Terminal inductance phase to phase (mH)	2.5
Torque constant (mNm/A)	217
Speed constant (rpm/V)	44.0
Rotor inertia (gcm ²)	3060

When modelling the motor in later points of this project, it will be considered as a usual DC motor to simplify the modelling, but as long as the current measured by the ESCON power stage is the current in one phase, the electrical torque can be estimated, as stated in [26], knowing that the three phases are symmetrical as follows:

$$\tau_m(t) = 2K_a i_p(t) \quad (2.1)$$

Which will be used later to identify the friction models containing the Striebeck curve from current and velocity measurements.

2.2.2 Encoder headers and codewheels

To integrate the Harmonic Drive transmission in a robotic system it is necessary to know the kinematic error variation during the rotation, for this task, an encoder is implemented, on each side of the gear. The encoders are sensors that generate digital signals according to a rotational movement, more precisely the incremental encoders are the ones used for the rotational movement. This kind of encoder determines its position counting pulses generated by a light passing through the holes of the codewheel attached to the rotational axis which wants to be measured. For determining the direction of the rotation two channels are used to measure the phase respect each other to know which direction is moving towards.



Figure 9. Encoder and codewheel HEDS-9100.

The incremental encoders channel A and B give two square waves with 90° phase between them. Sometimes a third channel called Index is implemented to know when the axis moved 360° ; this channel is usually used for initialization of the motors, also called Homing. In this project, some Homing functions have been implemented, but due to the bad state of the codewheels, the interruptions generated by the index channel triggered more than once per revolution, so it wasn't working properly.

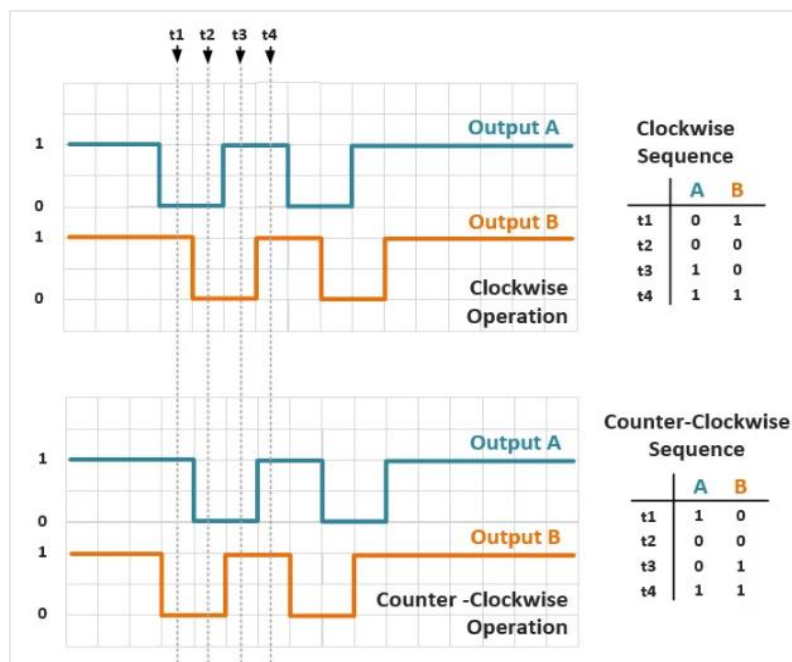


Figure 10. Encoder channels signal.

The given encoders use 2000 pulses per revolution so it's equivalent in radians can be calculated in the code when extracting the data for identification. In addition, the counter of the microcontroller has 16 bits, so each time the encoder readings reach this maximum the value is resetted, this is known as wrap or auto-reload, all this will be shown in more detail in the system limitations section.

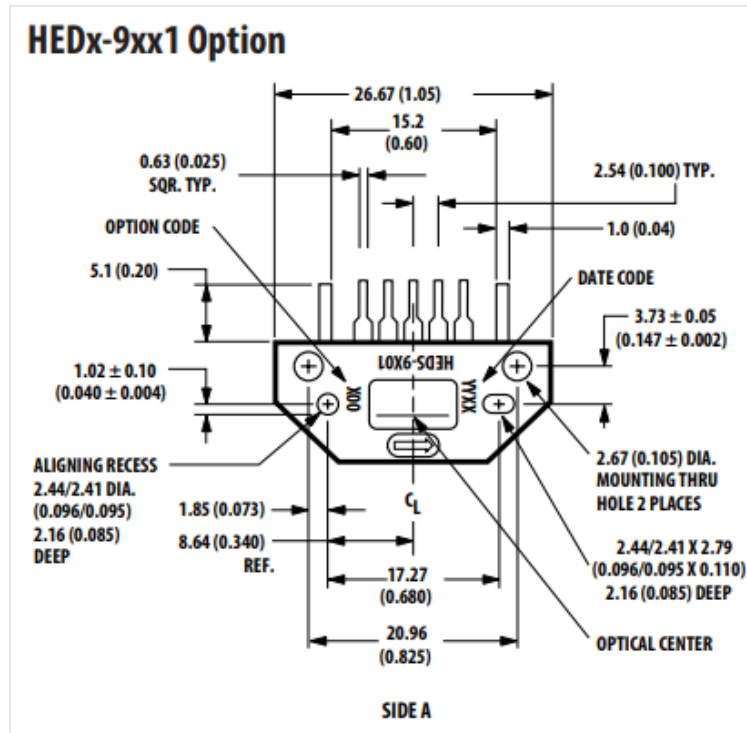


Figure 11. Encoder header dimensions.

The same encoder header and codewheel has been used in both motor and link side measurements, as it will be seen later, this has become one of the main system limitations when identifying the H.D flexibility parameters, as long as this parameters give the main information about the link side movement. The H.D reduces by 70 the velocity, this means that compared to the motor velocity, it will move very slowly so the quantization error will appear, then when estimating the velocity with signal differentiation from this encoder measurements will provide an inaccurate signal measured which will make the identification of the experimental platform a complex challenge.

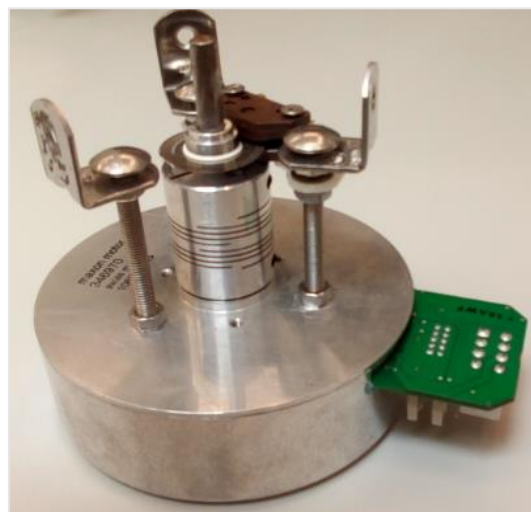


Figure 12. BLDC Motor with encoder header and codewheel mounted.

2.2.3 Discovery STM32F429Zi

The needs of this project at the hardware level requires, mainly, encoder readings, actuator signals, and data storage and transmission. For this purpose, the Discovery STM32F429Zi has been used in the project. Some of its more important features for the project are GPIOs, timers, DACs, SDRAM and USB VCP.

The GPIOs are used for the encoder readings, each channel of the encoder is connected to a GPIO pin, which is configured and attached to a timer, there is an encoder mode when configuring the GPIO in the code, so the counts are stored in a timer and then the value is read from the counter of that timer. The DAC value range is from 0V to 3.3V by specification, in reality this value may vary between devices. When configuring the DAC it is assigned to a pin in the board, the user can also set the speed or wave generator mode if needed; in this case, the signal will be generated by an equation depending on the time of the experiment.

The experiments are run with a control timer, which is configured to generate an interruption each millisecond that will be the sample time of the data measured. All the data collected will be transmitted via USB with the VCP, the virtual com port allows, when installed the proper drivers, to acquire data in the computer by connecting it to the Discovery board via USB.

The discovery board also has its own supply pins, which will be used to supply the encoder headers at +5Vdc.

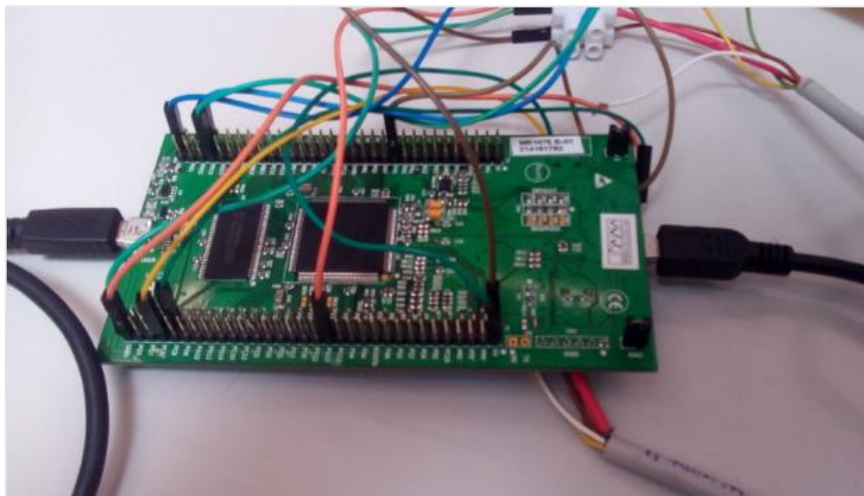


Figure 13. Discovery STM32F429Zi.

In this case, the C program consists in a Setup and a While, the Setup configures all the functionalities of the board used and this sequential way of working is simple and good when all the tasks have similar priority.

The STM32F4 has two DAC channels (12 bit) that can be used to send analog signals to the ESCON, which are implemented in the code, the ST-LINK V2, and VCP COM port drivers have to be installed in

order to load the programs to the board and send the data via USB both. In the table below all the pins used for the project are listed with its corresponding functionality.

Table 3. Discovery board pins used.

PIN	Function
GPIOC P6	Encoder 1 CHA
GPIOC P7	Encoder 1 CHB
GPIOD P12	Encoder 2 CHA
GPIOD P13	Encoder 2 CHB
GPIOA P5	DAC output
GPIOG P15	ESCON Enable digital signal
Vcc (+5Vdc)	Encoder 1 header
Vcc (+5Vdc)	Encoder 2 header
GND	Ground Encoder/DAC circuit

Remark that there are different pins used as GND but they are all internally connected inside the board.

2.2.4 EPOS/ESCON modules

When started, this project was using as a power stage for the motors the EPOS 24/5 positioning controller. However once the project is finished and the robot arm is placed in a vest where all the electronic components have to be fitted, the size is an important factor as well as the weight, if we add that the ESCON module can also work with the motors given for the application needed at lower price. Then it is obvious that the ESCON modules are more appropriate to be used in this project instead the EPOS, in this section, the ESCON modules are described and its operating modes are explained as well as some of its configurations used in the present project.

The ESCON Module 50/5 is a small sized servo controller for the highly efficient control of BLDC motors up to approx. 250 Watts.

The operating modes available are:

- Speed control (Closed loop).
- Speed control (Open loop).
- Current control.

For the identification procedure, only the speed control in open loop as well as the current control will be used. In the first, the DAC signal from the discovery is scaled to the voltage drop in the armature of the motor, when actuating over the voltage as input, the angular speed of the motor as output is changing. When in current control mode, the DAC signal is proportional to the current sent to the motor, thus, the electrical torque generated by the motor is controlled. The main problem of this, as it will be seen in later points is that in current control mode the voltage saturates, this means that the motor will reach its maximum speed in this control mode and so the data taken at saturating levels

cannot be used for the identification tasks. This is the main reason why voltage control experiments have been carried out in order to identify the parameters of the whole system. The ESCON Module 50/5 is designed to be commanded by an analog set value (DAC signal).

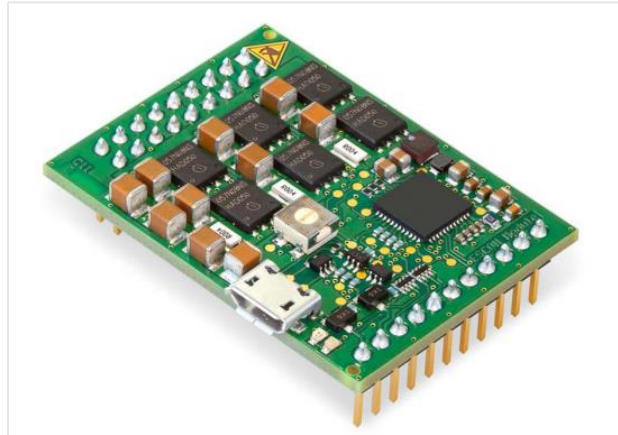


Figure 14. ESCON module 50/5.

This device is designed to be configure via USB interface using the graphical user interface “ESCON Studio” for windows PCs.

Some of its electrical rating properties as well as pin I/O used are shown in the tables below.

Table 4. Electrical rating properties.

Nominal operating voltage +Vcc	10-50 Vdc
Output voltage (max)	0.98 x + Vcc
Output current Icont/I _{max} (<20 s)	5A/15A
Max speed EC motor	150000 rpm

Table 5. Inputs and outputs used.

Analog input 1/2	12bit resolution; -10...+10V; differential
Hall sensor signals	H1, H2, H3
Digital input	+2.4...+36 Vdc (R _i =38.5 kOhm)
EC motor connection	Motor winding 1, 2 and 3
Status indicator Operation/Error	Green/Red Led

The supply has been set to 36V, which is the maximum the motor can use, this was done as long as the load is not known, otherwise the supply voltage should be calculated depending on the load. Note that the necessary output current is depending on the load torque.

All the pins available in the ESCON are shown in the figure below.

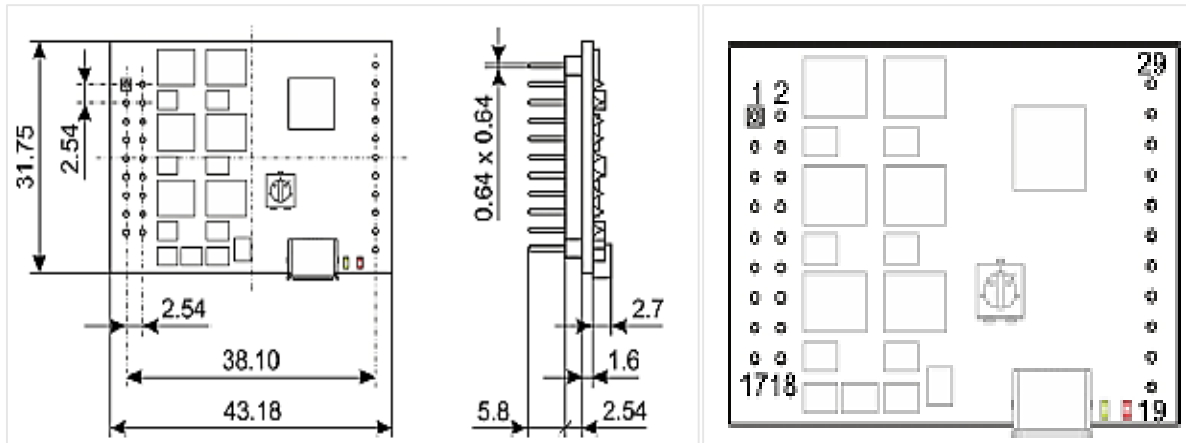


Figure 15. ESCON module 50/5 dimensions and pin assignment.

In the Table 6, the pins used are mentioned with its corresponding functionality described.

Table 6. Pin assignment table.

Pin	Signal	Description
1/2	Motor winding 1	EC motor winding 1
3/4	Motor winding 2	EC motor winding 2
5/6	Motor winding 3	EC motor winding 3
7/8	+Vcc	Nominal operating voltage
9/10	Power_GND/GND	Vcc GND/GND
11	+5Vdc	Hall sensor supply
13	Hall sensor 1	Hall sensor 1 input
15	Hall sensor 2	Hall sensor 2 input
17	Hall sensor 3	Hall sensor 3 input
21	DigIN2	Digital input 2
23	GND	Ground
26	AnIN2-	Analog input 2, negative signal
27	AnIN2+	Analog input 2, positive signal

In the next point the cables used for the ESCON supply, motor windings and Hall sensors are described. The analog input is the DAC signal amplified to adapt the DAC voltage range from [0V - 3.3V] to [0V-5V], also the before mentioned ENABLE signal corresponds to the Digital Input 2 which is configured

to activate the ESCON while the experiment is running and turn it off when the experiment ends for safety reasons.

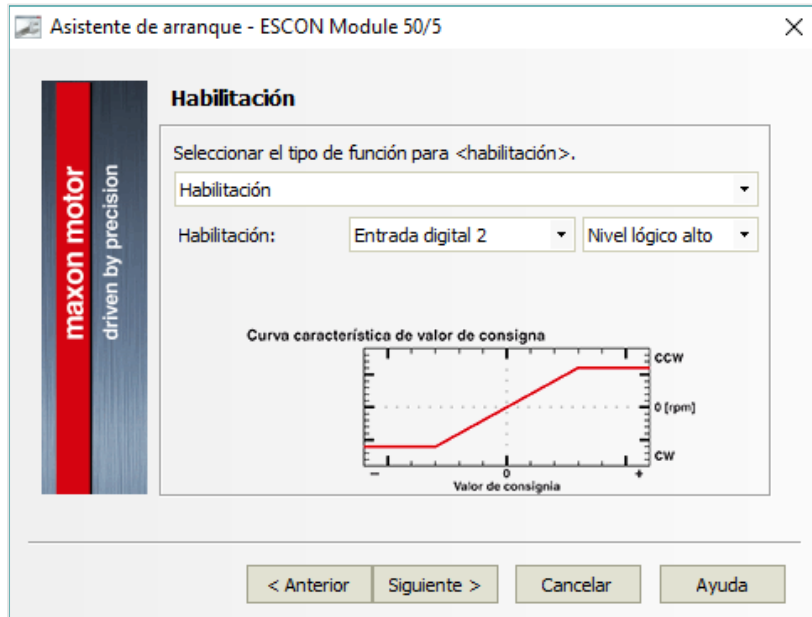


Figure 16. Enable pin configuration in ESCON Studio.

The ESCON studio is the software provided by the manufacturer of the power stage module and the BLDC motor. When configuring the power stage the mode that the motor is being actuated by the ESCON power stage depending on the analog signal received from the Discovery DAC is configured, here the different modes of control used to identify are explained. All the motor parameters have to be introduced into the ESCON studio, the steps are the following:

1. Choose type of motor (EC/BLDC or DC).

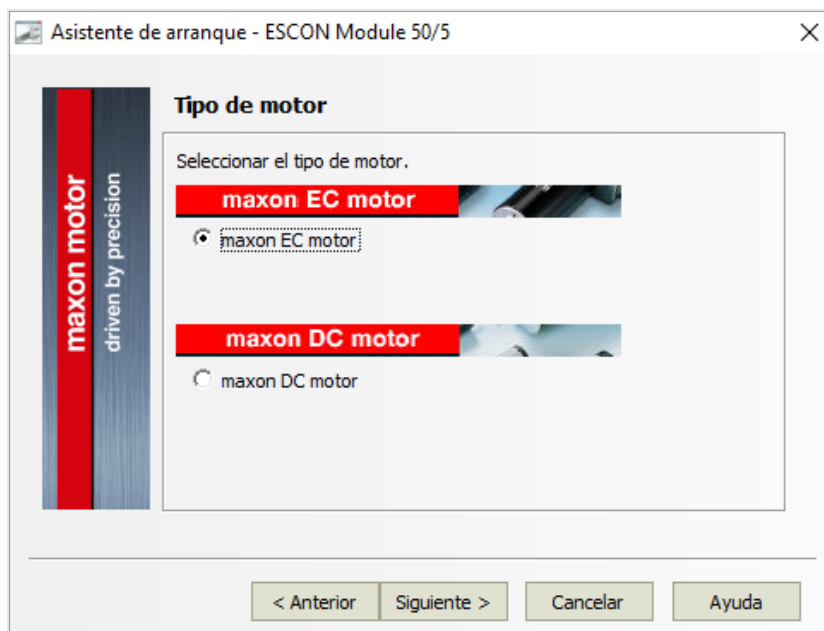


Figure 17. Motor type select.

2. Motor data specified by the manufacturer.

Asistente de arranque - ESCON Module 50/5

Datos del motor

Introducir características motor (más detalles, consultar catál. maxon).

Constante de velocidad:	44.0 rpm/V
Constante térmica de tiempo del devanado:	60.0 s
Número de pares de polos:	12

< Anterior Siguiete > Cancelar Ayuda

Figure 18. Motor parameters from datasheet.

3. System operating conditions (boundaries).

Asistente de arranque - ESCON Module 50/5

Datos del sistema

Introducir los datos del sistema.

Velocidad límite:	1280.0 rpm
Intensidad nominal:	1.7800 A
Máx. intensidad de salida:	3.5000 A

< Anterior Siguiete > Cancelar Ayuda

Figure 19. Motor operating point specified in datasheet.

4. Hall sensor polarity.

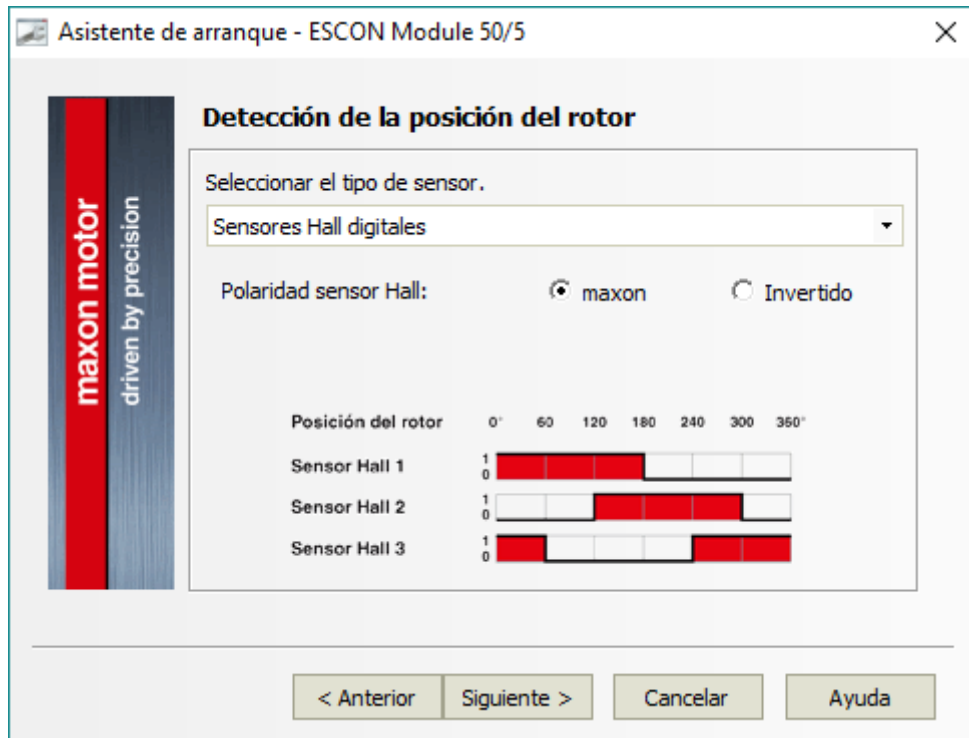


Figure 20. Hall sensor pattern configuration.

Once these steps are completed, the following step is to choose the operating mode; this is basically define if the DAC input will control the current in the motor or the voltage (speed). Both operating modes are explained in the following section.

Current control

In this mode, the current is proportional to the DAC signal, this means if the DAC is a sinus wave, the current will have the same shape but scaled by a factor, which is defined in the ESCON studio software, the diagram of this operating mode is the following.

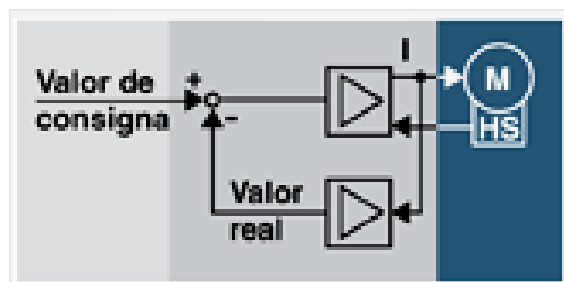


Figure 21. Current control ESCON diagram.

Therefore, to keep the current signal following the reference, the real current measured in the motor phase is feedbacked and compared to the set value. In the following figure, the effect can be appreciated.

Modelling, identification and control of an elastic joint.

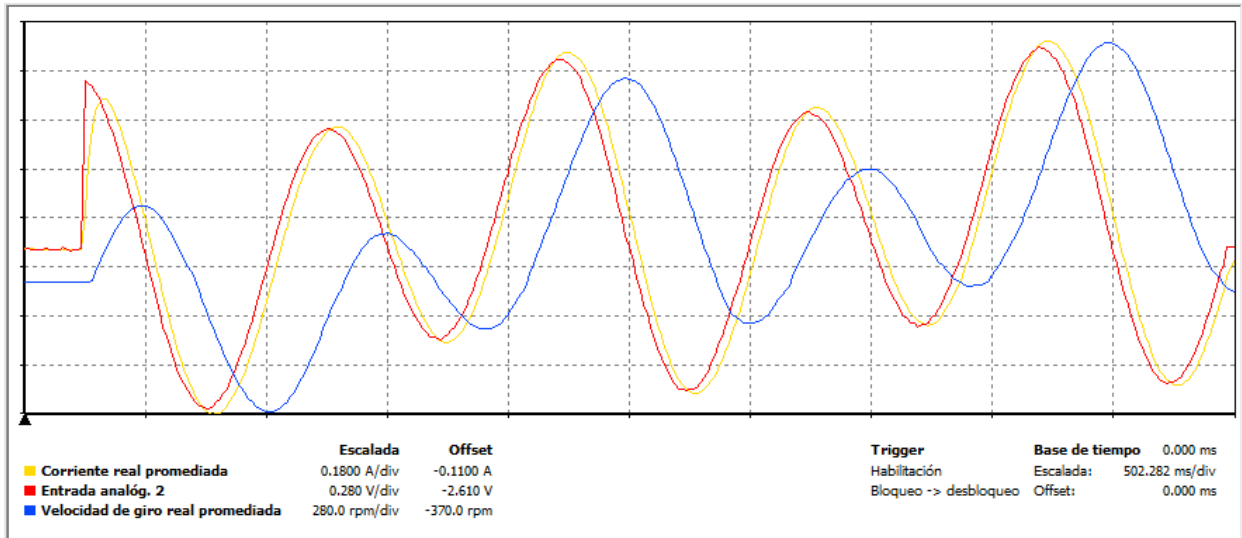


Figure 22. DAC input, Current averaged and motor speed in the ESCON data recorder.

The yellow signal corresponds to the current measured in one phase of the motor, the red one corresponds to the DAC input, in current control, the current follows the DAC signal, while in voltage control the speed signal follows the DAC input.

Define the set point depending on the working range of the input and output.

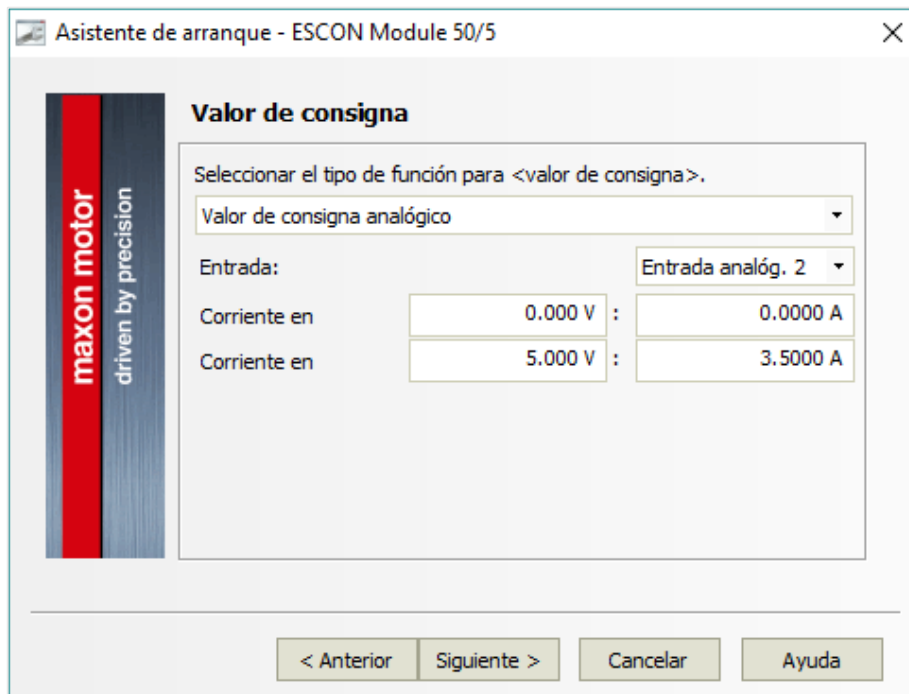


Figure 23. Set point configuration for current control.

Apply an offset, in Amperes, for moving the motor in both directions.

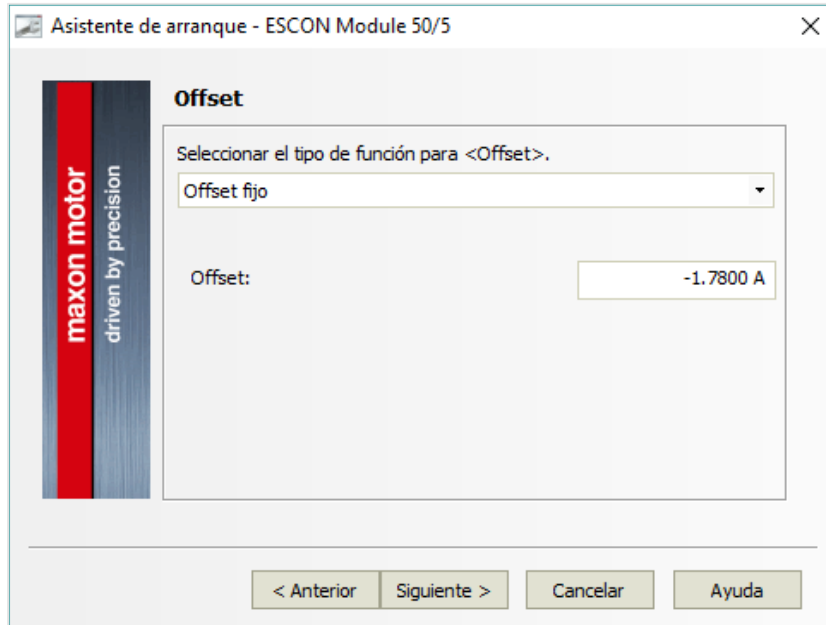


Figure 24. Offset needed for operating in two directions

If the configuration is as shown in figure 23 and 24, at 0V the power stage will deliver 0A. When adding the negative offset, that is the half of the maximum value we set as a maximum for the current output, we get the symmetric range for positive and negative sides. With 2.5V being the motor stopped, and the values from 0V to 2.5V negative direction and 2.5V to 5V positive direction.

In the ESCON Studio the values for the operating loop can be monitored on-line, and some values can be modified if needed, these are the offset and the power stage Gain.

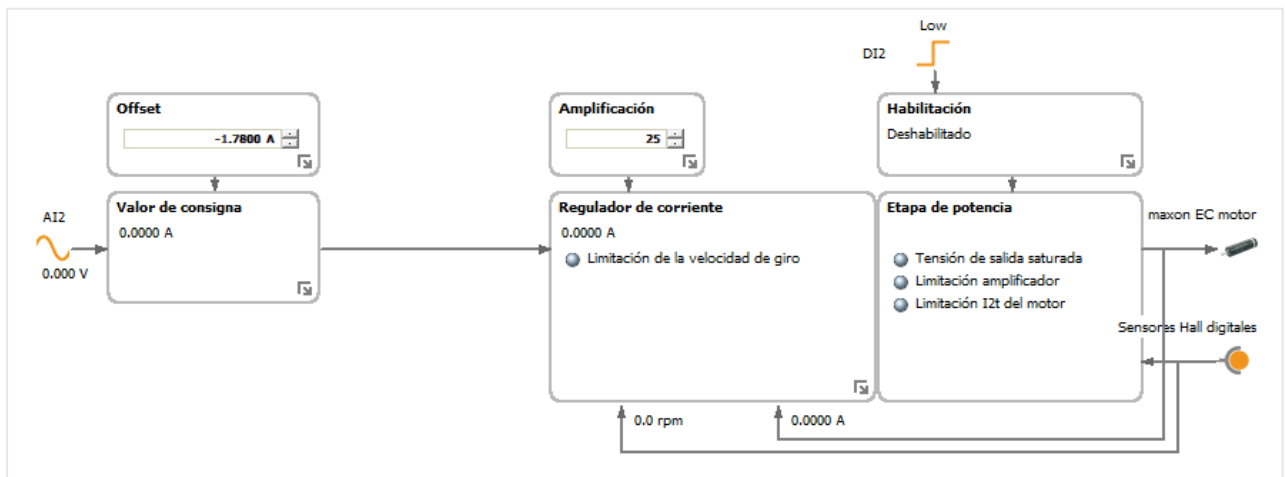


Figure 25. Current control ESCON monitored.

In the case of the gain tuning, there is an assistant where, when specifying the current and speed values for the motor, it calculates the gain needed in order to meet the requirements specified.

For achieving this, the motor will move, so a notification will appear for safety reasons before starting the regulator tuning.

In the figure below, the maximum current in amperes and the maximum speed in revolutions per minute are specified, then click the button init and let the motor move for the regulator tuning.

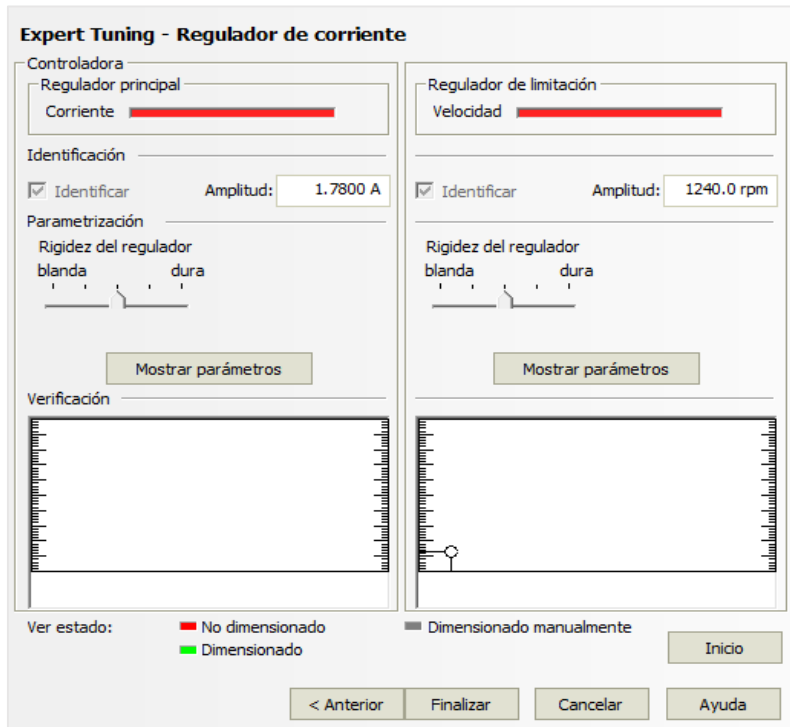


Figure 26. Proportional current controller tuning.

Voltage control

The voltage control, as can be seen in the figure below actuates on the voltage of the motor armature, here the back EMF acts, being the rotational speed times the motor speed constant subtracted to the voltage in the armature.

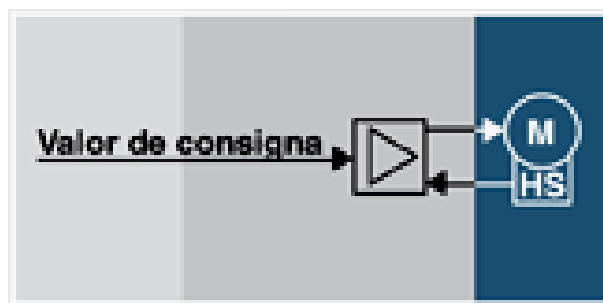


Figure 27. Voltage control diagram.

Define the set point depending on the working range of the input and output.

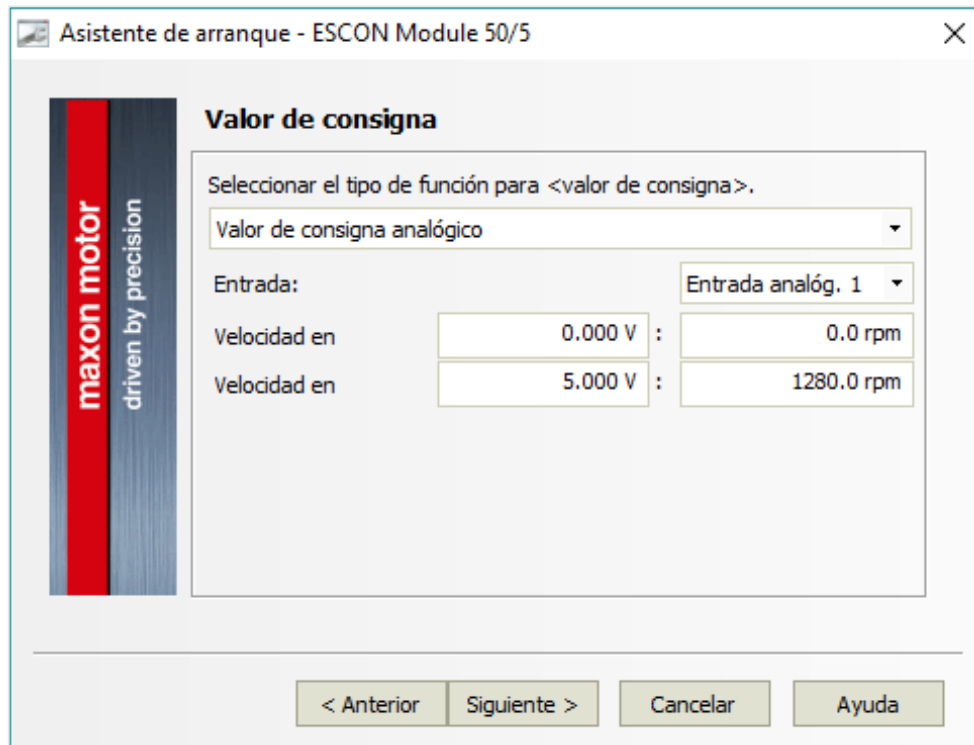


Figure 28. Power stage gain definition.

When selecting the set point value, the upper and lower values have to be specified, this means that the value of the maximum speed of the motor corresponds to the maximum value on the ESCON analog input. In the figure shown the system was set to operate only in one direction, this means 0V corresponds to the motor stop and 5V corresponds to the maximum value the motor can rotate.

When configuring the power stage for controlling the voltage in two directions, the maximum value of the speed has to be doubled, then an offset has to be set equal to the maximum value but negative.

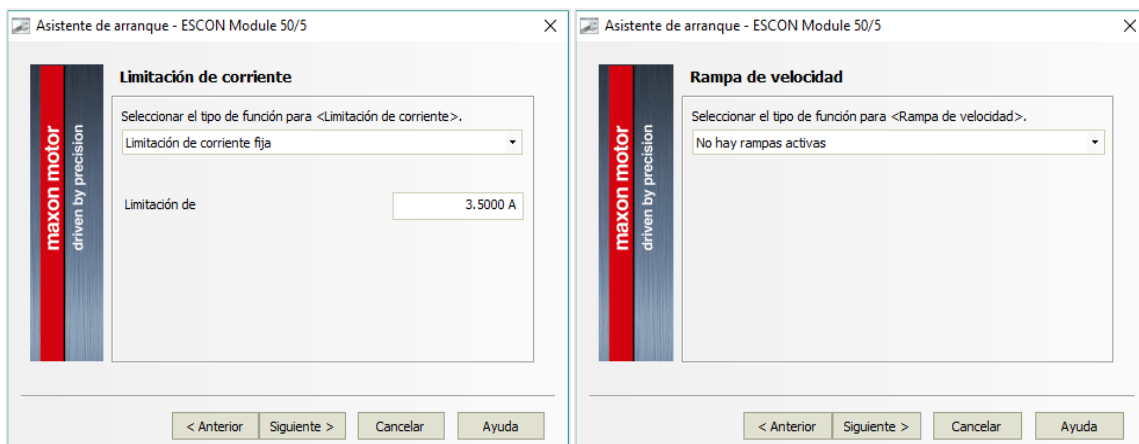


Figure 29. Current limitation and speed control ramp configuration.

When working in voltage control, the offset has to be specified in rpm's in order to move the motor in both directions.

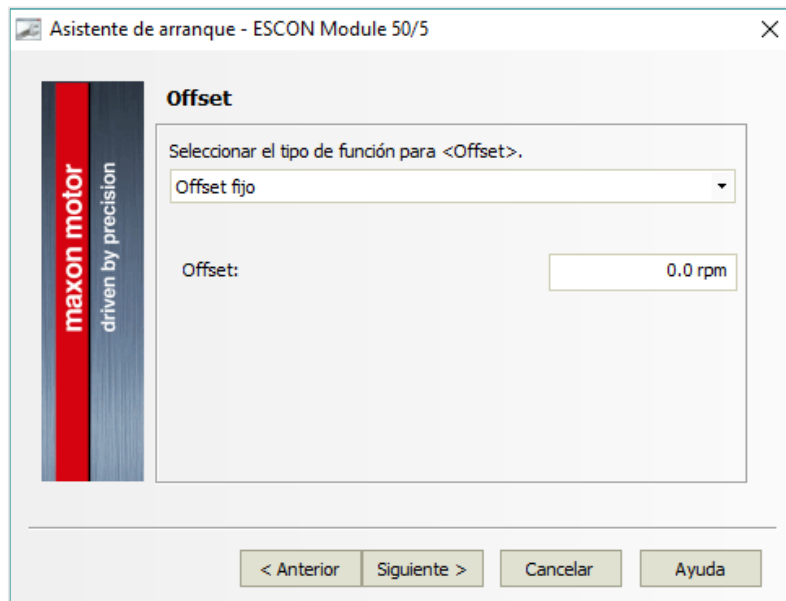


Figure 30. Speed control offset.

In a similar way, compared to current control, the speed control can be monitorized, in this case as long as the voltage can be varied precisely depending on the DAC input, no regulation loop is needed in order to follow the signal demand, so no regulation tuning is needed.

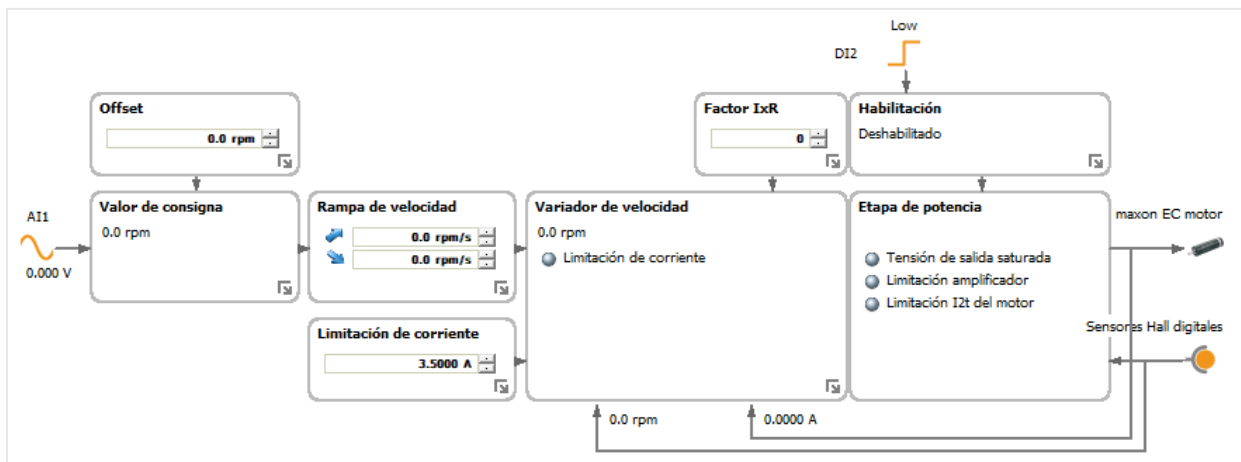


Figure 31. ESCON studio speed control (open loop).

In the ESCON Studio the analog input value, this is the voltage of the DAC multiplied by the amplifier gain 5/3 can be monitored. At this point the DAC1 assigned to PA4 in the STM32F429 Discovery had an offset error, this means that when 0 volts are required as DAC output, the Discovery provides constant 0.12V instead 0V, known this the DAC2 in PA5 is tested and there is no such offset. When implementing the system the DAC calibration has to be a primary thing, but for the low cost platform in which this project is based is enough.

2.2.5 Wires

In this section the wires used to connect the motor to the power stage ESCON will be explained, the motor cable and the hall sensor cable are shielded, the main reason is to reduce the electrical noise from affecting the signals and to reduce the electromagnetic radiation that may interfere with other devices. Mainly three kind of cables are needed to connect the BLDC motor to the power stage, these are:

- Power supply cable.
- Motor power cable.
- Hall sensor cable.

The particular shape of the back EMF in the BLDC motors makes necessary to know the electrical degrees of the motor which is recorded by the hall sensor, this means that the ESCON won't work until it detects the Hall sensor cable is connected and working properly.

Table 7. ESCON power cable information.

ESCON Power cable technical data and pin assignment	
Cable cross-section	2 x 0.75mm ²
Head A	Cable connected to the supply
Head B	Cable sleeves 0.75 mm ² , Soldered to the ESCON
Wire Black Head A pin 1 (-)	Ground of supply voltage
Wire Black Head A pin 2 (+)	Supply voltage +11...+36 Vdc



Figure 32. ESCON power supply cable.

Modelling, identification and control of an elastic joint.

Table 8. ESCON motor cable information.

ESCON Motor cable technical data and pin assignment	
Cable cross-section	3 x 0.75 mm ² shielded
Head A	Molex Micro-Fit 4P female crimp terminals (Motor)
Head B	Cable sleeves 0.75 mm ² , Soldered to the ESCON Pins
Wire White Head A pin 1	BLDC Motor: Winding 1
Wire Brown Head A pin 2	BLDC Motor: Winding 2
Wire Green Head A pin 3	BLDC Motor: Winding 3
Wire Black Head A pin 4	Cable shield



Figure 33. ESCON motor cable.

Table 9. Hall sensor cable information.

Hall sensor cable technical data and pin assignment	
Cable cross-section	5 x 0.14 mm ² shielded
Head A	Molex Micro-Fit 6P female crimp terminals (Motor)
Head B	Cable sleeves 0.75 mm ² , Soldered to the (ESCON) Pins
Wire Green Head A pin 1	Hall sensor 1
Wire Brown Head A pin 2	Hall sensor 2
Wire White Head A pin 3	Hall sensor 3
Wire Yellow Head A pin 4	GND
Wire Grey Head A pin 5	Hall sensor supply voltage +5Vdc
Wire Black Head A pin 6	Hall shield

Modelling, identification and control of an elastic joint.



Figure 34. Hall sensor cable.

In the following picture these connections are shown for the actual assembly of the power stage.

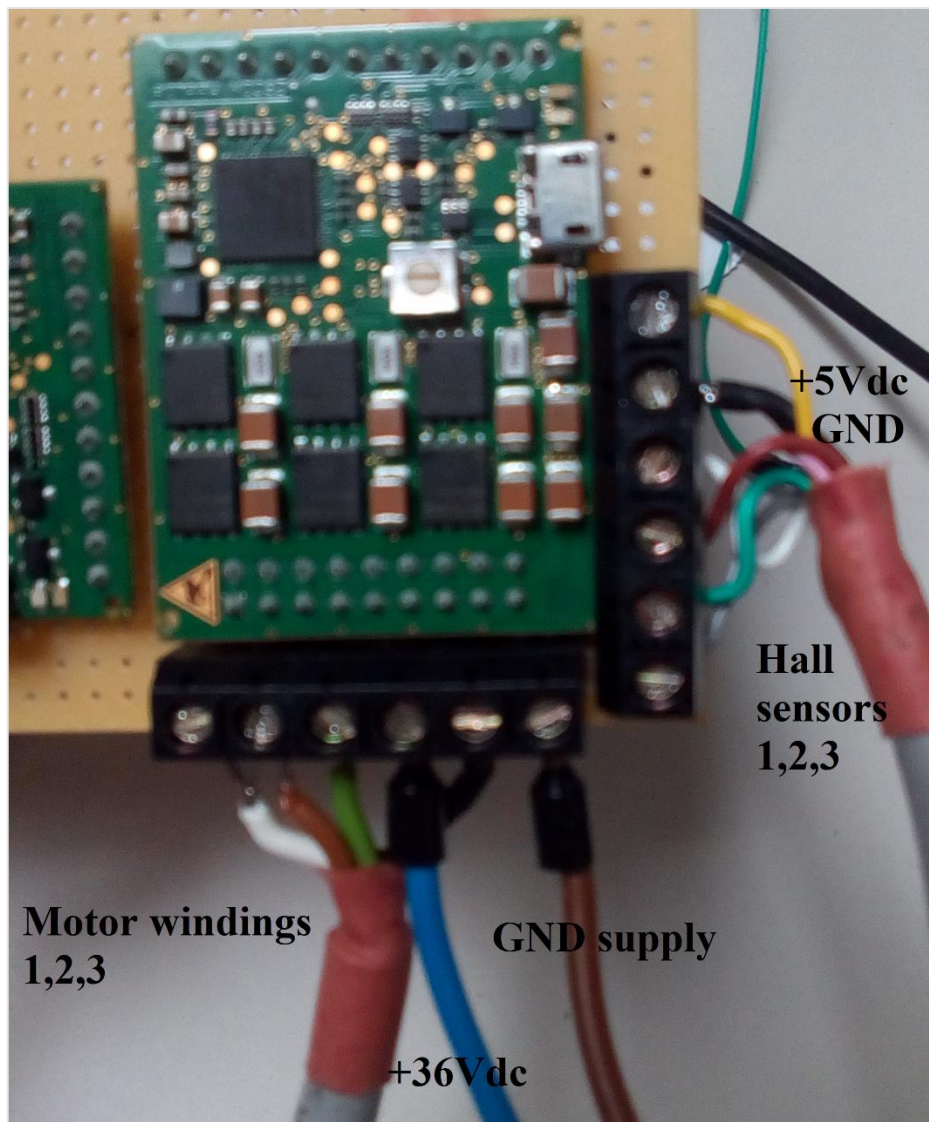


Figure 35. ESCON Module and the motor cables.

2.2.6 Auxiliary electronic circuits

Signal conditioning from the Discovery DAC (0-3V) to the ESCON input (0-5V) Battery for the autonomous system.

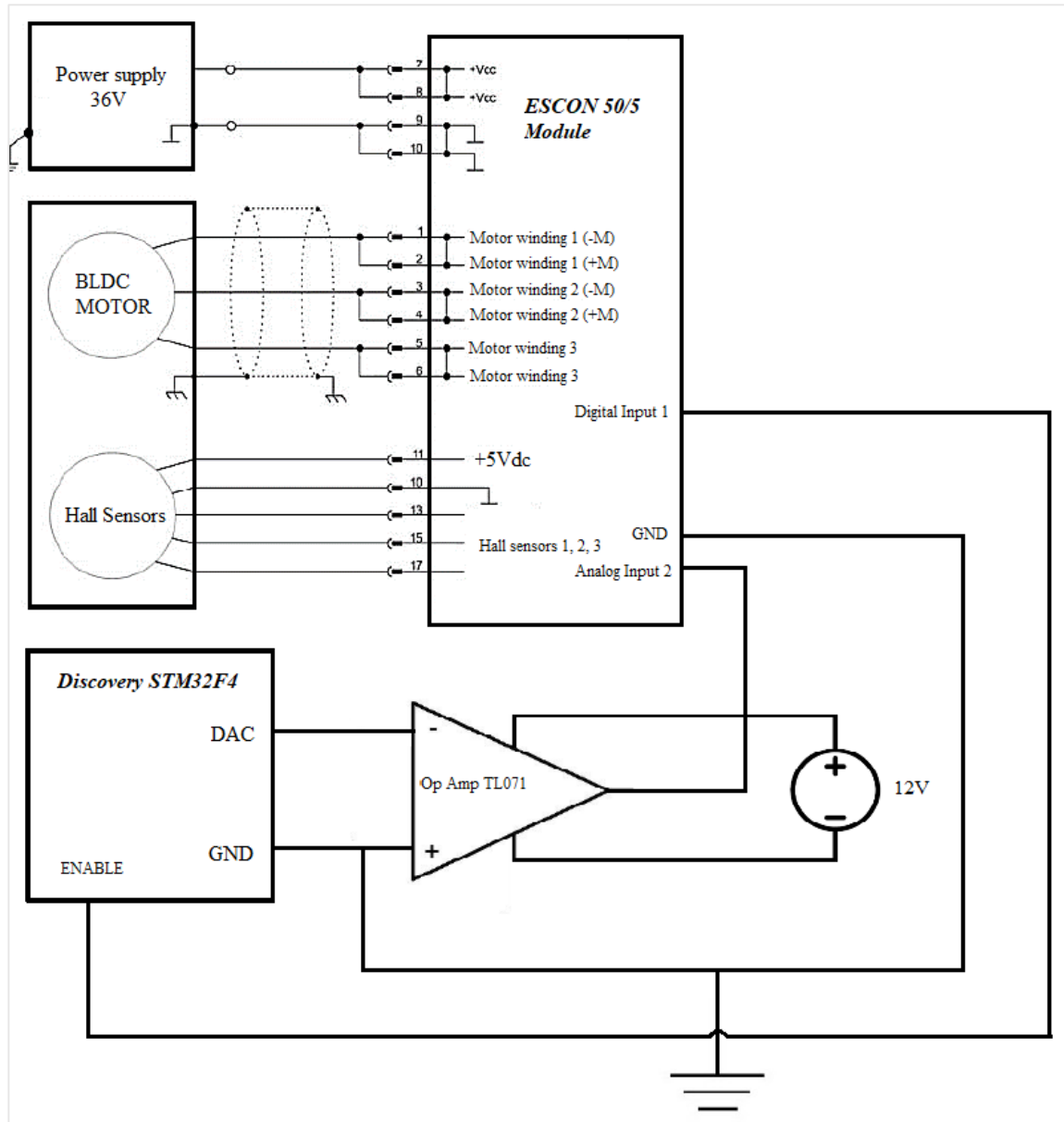


Figure 36. System connections diagram.

Remark that this would be the general diagram for one DAC into the ESCON power stage. When mounting as many ESCONs as motors are needed, this diagram should be repeated if the Discovery DAC is used as analog signal to demand the current or voltage needed in order to move the motor.

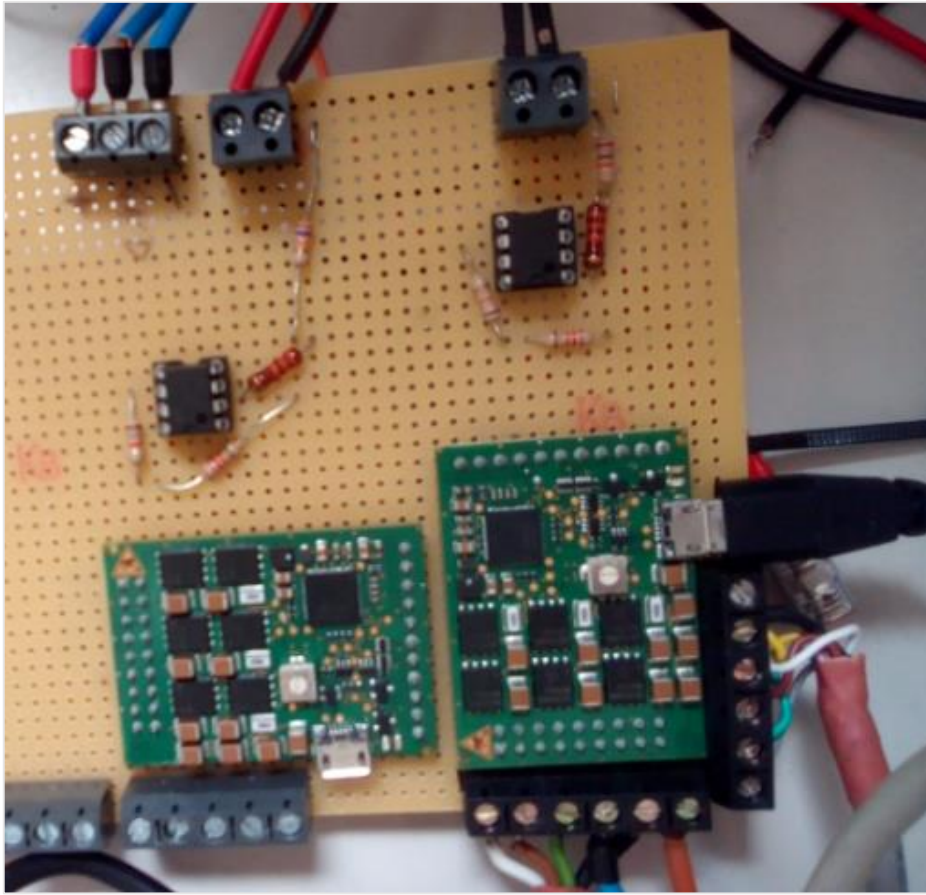


Figure 37. Electronic circuit in the power stage.

2.2.7 System architecture

The system used in this project is a part of a bigger system, where the robot arm will be controlled with a Raspberry pi, and many Discoveries will be used to access to low level hardware, this means all the encoders for the robot arm, all the DACs for sending the actuator signal and communication between dispositives.

In the project, the Discovery is connected to the PC with the ST-Link USB and the VCP USB (in case the experiments are carried out for taking data of the system), for using these appropriately, the ST-Link and VCP com port drivers are installed in the PC. The Harmonic Drive transmission is mounted into the EC motor, once assembled, it is mounted into the prototype robot structure. The encoders are connected from the headers built inside the structure to the Discovery. The DAC is connected to the auxiliary amplifier circuit, which amplifies the voltage range from 3V to 5V, then this voltage is the analog input to the ESCON, which is supplied by a 36V power source. The ESCON and the EC motor are connected through the cables explained in the section above. The ESCON requires the hall sensor signals to operate properly, so this connection cannot be avoided, also the three wires for the motor windings are connected, and the ESCON module is connected to the PC via USB and the ESCON studio software has to be installed into the computer in order to manage the operation conditions of the ESCON.

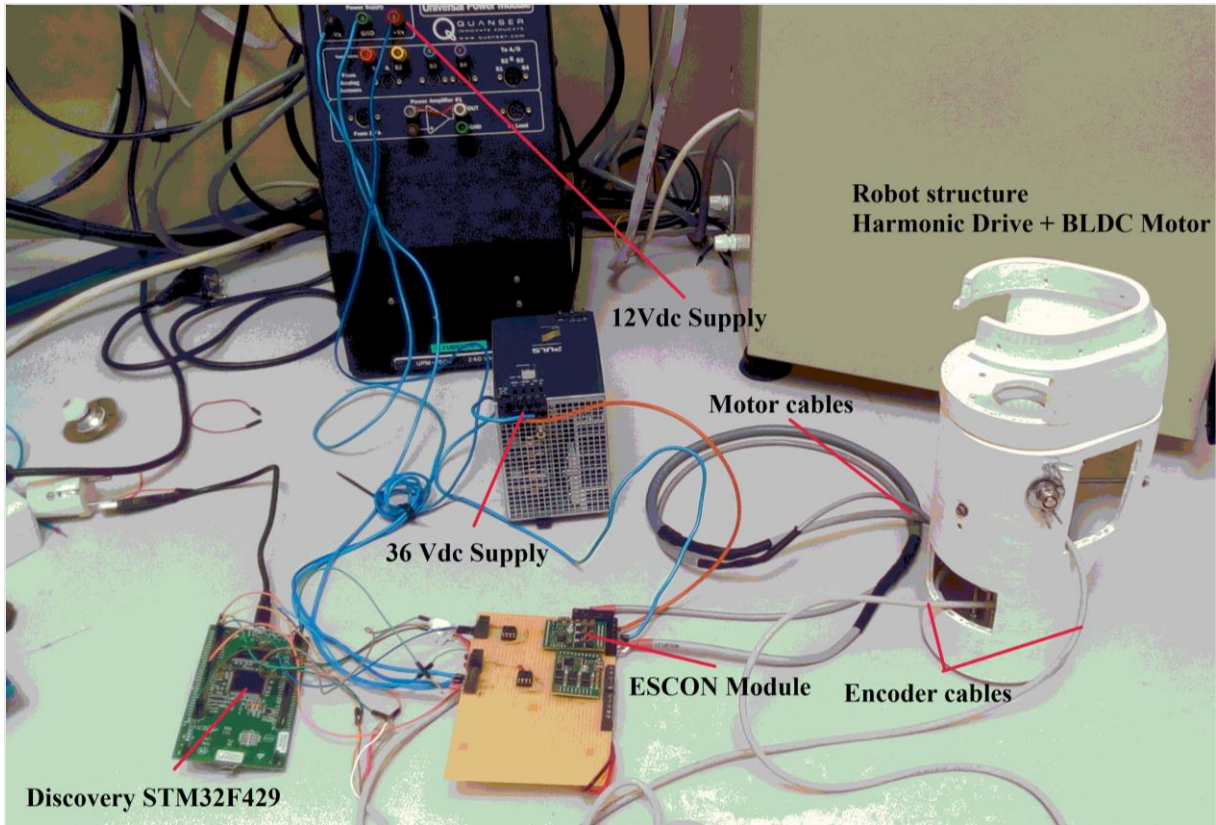


Figure 38. Experimental platform mounted in the DISA.

3. Modelling robots with flexible joints

When modelling robot dynamics, the usual assumption is that manipulators consist only of rigid bodies. However, this ideal situation may be considered valid only for slow motion and small interacting forces. If flexibility is not taken into account when considering robot design and control, a degradation of the overall expected performance of the robot occurs.

From a modelling point of view, flexibility can be assumed as concentrated at the robot joints, the dynamic modelling steps are similar to the rigid case, with the need to introduce additional generalized coordinate besides those used to describe the rigid motion of the robot arm.

The presence of joint flexibility in the present work is given by the use of Harmonic Drives as the transmission and reductor element, which, as stated before provides high reduction ratios with power-efficient compact inline devices.

When subject to the forces/torques arising in normal robot operation, this components are flexible, in this case the “flexspline” of the harmonic drive, this introduces a time varying displacement between the position of the actuators and that of the driven links. Without a specific control action, an oscillatory behaviour typically of small magnitude but at relatively high frequency, is observed at the robot-end effector level during free motion. In addition, some form of instability may occur in tasks involving contact with environment.

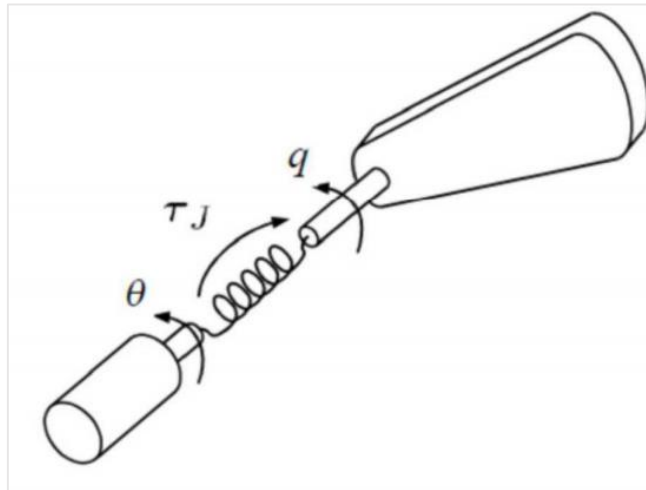


Figure 39. Elastic joint approach.

Dynamic models that include joint flexibility are used to evaluate quantitatively vibratory effects superposed on the rigid motion, for the multivariable case some assumptions are made for obtaining a simplified but still good dynamic model, but this is out of the scope of this project, and maybe will be part of posterior projects.

3.1 Modelling the Harmonic Drive transmission in a robot joint

Considering the topology of an elastic robot joint as shown in Figure 40. In terms of the input-output behaviour, the proposed structure does not differ from a simple fourth order linear dynamic model of two connected masses. An external exciting torque τ_m applied by the motor, which constitutes the input value. The relative position of the second moving mass q constitutes one of the outputs and its the angular position at the link side, θ is the motor side angular position, which is also measured, the derivatives of both signals are also in the model related to the non-conservative terms of the equations which are frictions and damping parameters.

The Lagrange equations, which is a method that allows writing down the dynamic equations of a general mechanical system from a generalized coordinates system, allows to represent the dynamic model as follows:

$$\frac{d}{dt} \left(\frac{\partial K_e}{\partial \dot{q}_i} \right) - \frac{\partial K_e}{\partial q_i} - F_i = 0 \quad (3.1)$$

Where:

q_i Generalized coordinates of the system or degrees of freedom.

K_e Kinetic energy of the system.

F_i Generalized force.

The generalized force term F_i considers all the forces interacting in the system, these forces can be of three different categories:

1. Interaction forces between different punctual masses of the system.
2. External forces, included gravity.
3. Friction forces.

Forces 1 and 2 are conservative and they can be obtained differentiating a potential energy function U with respect to a generalized coordinate, force 3 represent friction force which is non-conservative and its obtained deriving respect to the velocity a potential dependent of the velocity, which is the Rayleigh function ϑ . So the generalized force can be written as:

$$F_i = - \frac{\partial U}{\partial q_i} - \frac{\partial \vartheta}{\partial \dot{q}_i} \quad (3.2)$$

Where the generalized force is represented with its conservative and non-conservative terms.

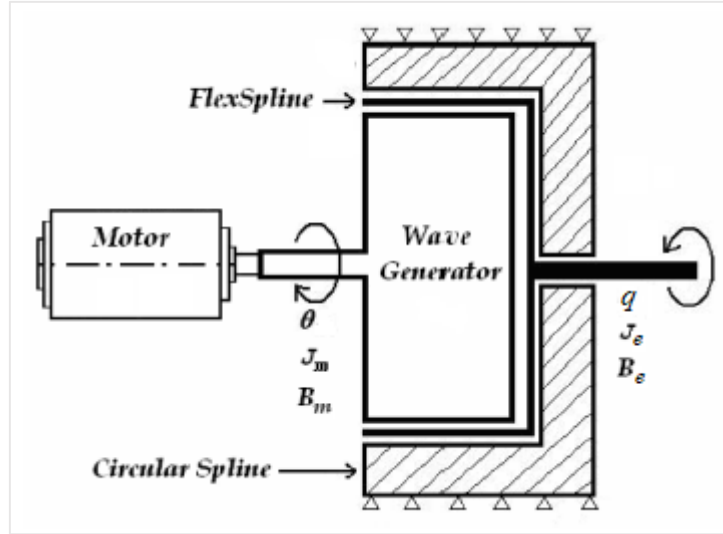


Figure 40. Harmonic Drive configuration used in the application.

Therefore, for the basic configuration shown in Figure 41, the Lagrange equations can be applied as a mechanical system with two D.O.F.

1. Position or angle of the motor θ_m , which is the input to the system.
2. Kinematic error $\Delta\theta$, which is expressed as a function of the output position of the H.D (θ_l) which is determined by experimental measurements and its calculated as:

$$q = \theta - \Delta\theta \quad (3.3)$$

The kinetic energy K_e it's due to the movement in the motor axis and in the output axis where the load is.

$$K_e = \frac{1}{2}J_m\dot{\theta}_m^2 + \frac{1}{2}J_e\dot{q}^2 \quad (3.4)$$

The elastic potential energy it's due to the flexibility of the mechanism, and it is defined as:

$$U = \int_{-\theta_s}^0 K_{el}\Delta\theta d\Delta\theta \quad (3.5)$$

The Rayleigh function is defined as:

$$\vartheta = \sum_{i=1}^n \frac{1}{2}B_i\dot{q}_i^2 \quad (3.6)$$

When applied to the friction terms the following expression is obtained:

$$\vartheta_f = \frac{1}{2}B_m\dot{\theta}_m^2 + \frac{1}{2}B_e\dot{q}^2 \quad (3.7)$$

In addition, the spring damping has to be included in this kind of function, as long as it is a dissipative term.

$$\vartheta_d = \frac{1}{2}B\Delta\dot{\theta}^2 \quad (3.8)$$

There is one Lagrange equation per each generalized coordinate, so for representing the H.D system 2 equations are needed.

$$\tau_m = \frac{d}{dt} \left(\frac{\partial K_e}{\partial \dot{\theta}_m} \right) - \frac{\partial K_e}{\partial \theta_m} + \frac{\partial U}{\partial \theta_m} + \frac{\partial \vartheta_f}{\partial \dot{\theta}_m} - \frac{\partial \vartheta_d}{\partial \dot{\theta}_m} \quad (3.9)$$

$$0 = \frac{d}{dt} \left(\frac{\partial K_e}{\partial \dot{q}} \right) - \frac{\partial K_e}{\partial q} + \frac{\partial U}{\partial q} + \frac{\partial \vartheta_f}{\partial \dot{q}} - \frac{\partial \vartheta_d}{\partial \dot{q}} \quad (3.10)$$

Including all the terms developed above, the two last equations can be written as:

$$J_m \ddot{\theta}_m - K\Delta\theta - B\Delta\dot{\theta} + B_m \dot{\theta}_m = \tau_m \quad (3.11)$$

$$J_e \ddot{q} + K\Delta\theta + B_e \dot{q} + B\Delta\dot{\theta} = 0 \quad (3.12)$$

Where:

J_e Link inertia moment.

b_e Link viscous friction.

B Spring damping.

K Linear spring stiffness coefficient.

J_m Motor inertia moment.

b_m Motor viscous friction.

For simplicity in the equations above, the reduction ration has been omitted, when the reduction ratio is taken into account, all the torques in the motor side of the equation are divided by N, so the equations (3.11) and (3.12) can be rewritten as:

$$\tau_m = J_m \ddot{\theta} + b_m \dot{\theta} + \frac{K}{N} \left(\frac{\theta}{N} - q \right) + \frac{B}{N} \left(\frac{\dot{\theta}}{N} - \dot{q} \right) \quad (3.13)$$

$$0 = J_e \ddot{q} + b_e \dot{q} + K \left(q - \frac{\theta}{N} \right) + B \left(\dot{q} - \frac{\dot{\theta}}{N} \right) \quad (3.14)$$

Here the input is the torque generated by the motor, which is proportional to the current in the motor armature, this means working in current control mode, as shown in the following block diagram.

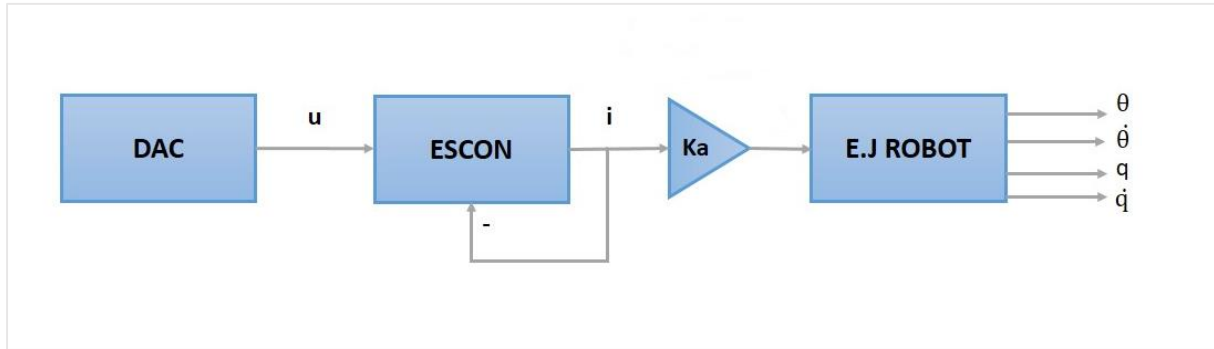


Figure 41. Current control block diagram.

The terms $\left(\frac{\theta}{N} - q\right)$ and its derivatives are the kinematic error and its derivative.

The input is the torque generated by the motor is calculated from the current, the motor current to torque constant gives the torque applied by the motor as follows:

$$\tau_m(t) = 2K_a i(t) \quad (3.15)$$

In this case, as long as there are two control modes available in the power stage (ESCON module) the formulas that include the input (motor action) can be rewritten as follows for voltage control:

$$\ddot{\theta}(t) = \frac{\left(u_{DAC}(t)K_{in} - \frac{\tau_c R}{K_a} \right) + \frac{\left(\frac{B}{N} \left(\dot{q}(t) - \frac{\dot{\theta}(t)}{N} \right) + \frac{K}{N} \left(q(t) - \frac{\theta(t)}{N} \right) \right) R}{K_a} - \left(K_w + \frac{B_m R}{K_a} \right) \dot{\theta}(t)}{J_m K_a / R} \quad (3.16)$$

With the following block diagram considered:

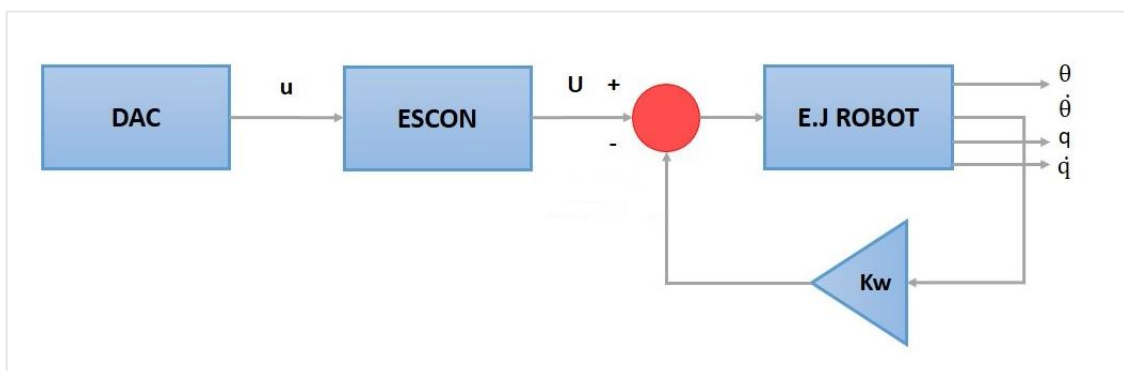


Figure 42 Voltage control block diagram

Where the back EMF acts as a “virtual friction” and its added to the viscous friction coefficient as in equation (3.16).

Despite this, when identified, the parameters from voltage control and the ones from current control will be separated.

3.2 Harmonic Drive state space modelling

The most usual way to study a system is through the relationship between the inputs given to the system and the outputs obtained from it. The system modelling in state space is based on describing the dynamic system by means of n first order differential equations or difference equations (for discrete systems), and then those equations can be represented in matrix form so the mathematical expressions are simplified.

A dynamic system, once the differential equations are known, the variables that define the system or the state variables are defined, then the equations are rearranged as a function of these and the state equations of the system are obtained

$$\dot{x}(t) = f[x(t), u(t)] \quad (3.17)$$

$$y(t) = g[x(t), u(t)] \quad (3.18)$$

Where:

$x(t)$ It's the state variables vector.

$\dot{x}(t)$ It's the derivative of the state variables vector.

$u(t)$ It's the input vector.

$y(t)$ It's the output vector.

If the system is linear and time invariant the state equations can be written as:

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (3.19)$$

$$y(t) = Cx(t) + Du(t) \quad (3.20)$$

We consider the following state space vector:

$$x = [\theta \quad \dot{\theta} \quad q \quad \dot{q}]^T \quad (3.21)$$

Rearranging the equations taking into account the spring damping:

$$J_m \dot{x}_4 + B_m x_4 + \frac{Kx_3}{N^2} - \frac{Kx_1}{N} + \frac{Bx_4}{N^2} - \frac{Bx_2}{N} = \tau_m \quad (3.22)$$

$$J_e \dot{x}_2 + b_e x_2 + Kx_1 - \frac{Kx_3}{N} + Bx_2 - \frac{Bx_4}{N} = 0 \quad (3.23)$$

From equations (3.19), (3.20) and (3.21) the following matrix are obtained which gives the state space representation of the given system.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{K}{N^2 J_m} & -\frac{b_m + \frac{B}{N^2}}{J_m} & \frac{K}{N J_m} & \frac{0}{N J_m} \\ 0 & 0 & 0 & 1 \\ \frac{K}{N J_e} & \frac{B}{N J_e} & -\frac{K}{J_e} & -(b_e + B)/J_e \end{bmatrix} \quad (3.24)$$

$$B = \begin{bmatrix} 0 \\ K_a K_m / J_m \\ 0 \\ 0 \end{bmatrix} \quad (3.25)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.26)$$

$$D = 0 \quad (3.27)$$

3.3 Modelling non-linearities

Nonlinear behaviour in elastic robot joints can be found by different phenomenas such as transmission compliance, resulting from gear tooth interaction and wave-generator deformation due to high radial forces that is commonly approximated by a nonlinear or piecewise linear stiffness. In addition, hysteresis torsion in elastic robot joints occurs as a coupled nonlinearity due to different factors as backlash, internal friction, and nonlinear stiffness. These interact inside of mechanical transmission assemblies. The nonlinear joint torsion leads to hysteresis lost motion and can provoke control errors in relation to the joint output at both trajectories tracking and positioning. In this chapter, the studied models includes the Bouc–Wen-like hysteresis model, together with nonlinear cubic polynomial for elasticity, which is originated from structural mechanics, both arranged according to the assumed torque transmitting structure.

Elasticities in robotic joints always have drawn an attention in the research since being one of the key challenges for an accurate joint dynamics modeling and control.

To this end it can be emphasized that evermore lightweight design co-determine the development of modern robotics as well, where evermore nonlinearities constitute a growing challenge.

3.3.1 Hysteresis spring

The input angle θ is converted into the harmonic drive output angle by the reduction ratio N , where the reductor deformation can be observed through $\Delta\theta$ as equation (3).

Where the “flexspline” rotation, q has an opposite direction respect to the “wave generator” θ .

Instead using the typical linear relation ($K(\Delta\theta)$), the transmission acting between the input and output is approximated by a cubic polynomial, which is function of the kinematic error as follows:

$$T_e(\Delta\theta) = K_1(\Delta\theta) + K_3(\Delta\theta)^3 \quad (3.28)$$

Where K_1 and K_3 are the cubic stiffness coefficients.

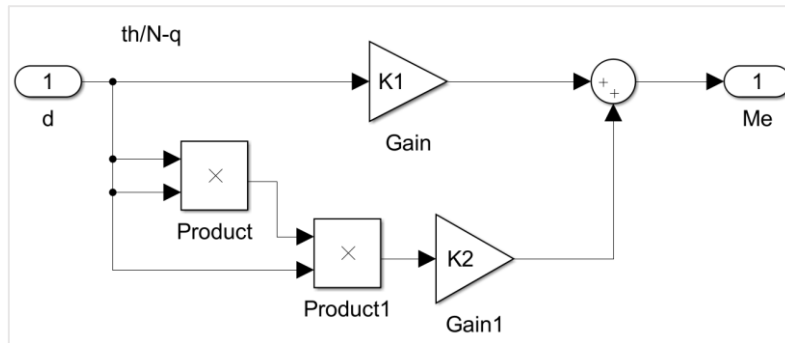


Figure 43. Nonlinear elasticity (cubic polynomial) Simulink implementation.

The torsional stiffness curve can be also approximated by three straight lines having stiffness of K_1 , K_2 and K_3 , but in the present work a cubic polynomial with two coefficients is used instead as long as provides good enough results.

Stiffness K_1 applies for output torque of zero to T_1 and stiffness K_3 applies for output torque greater than T_2 , stiffness K_2 applies for output torque between T_1 and T_2 .

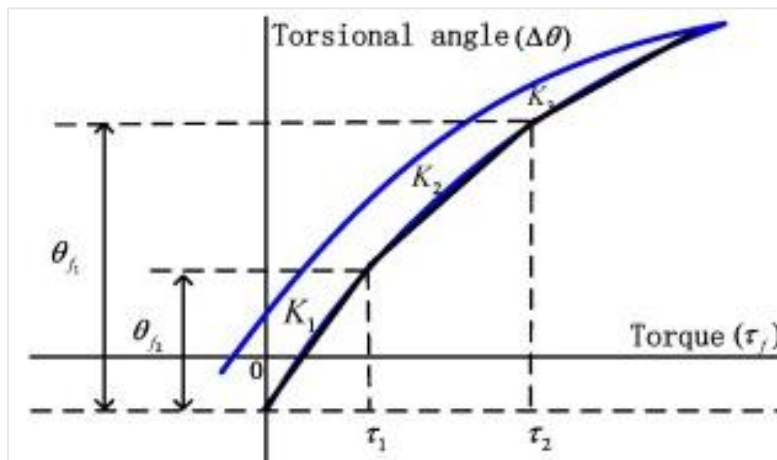


Figure 44. Non-linear elastic polynomial.

The fact of using this kind of function is justified for using it together with the Bouc-Wen hysteresis model, which contemplate the plastic behaviour of the transmission, then putting together the plastic-elastic physical phenomena in the same dynamic equation.

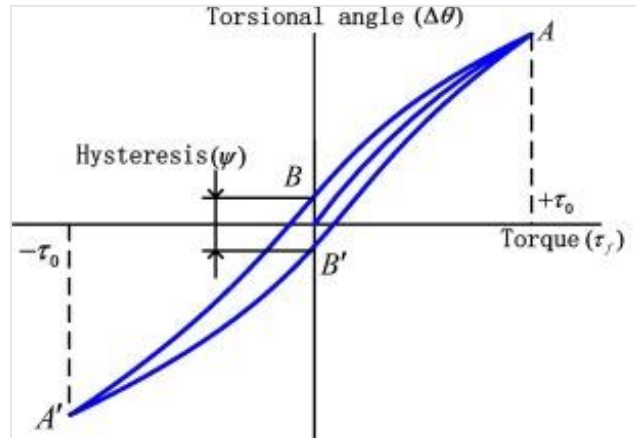


Figure 45. Typical hysteresis shape in Harmonic Drives.

The Bouc-Wen hysteresis model uses a first-order non-linear differential equation that relates in hysteretic way the relative displacement to the restoring force. In this equation, the shape of the hysteresis curve is modified by a set of free parameters, the main advantage of the Bouc-Wen hysteresis model is the possibility to include the complex non-linear stiffness characteristics exposed above. The restoring torque, which is a hysteresis spring, can be decomposed in an elastic and plastic term, a weighted of both determines the reversible and irreversible contribution to the overall transmitted torque value, and it is represented in the following equation:

$$Th(\Delta\theta, t) = wT_e(\Delta\theta)|\Delta\theta(t)| + (1 - w)T_e(\Delta\theta)|x(t)| \quad (3.29)$$

The weighting factor w provides the relation between purely elastic ($w = 1$) or purely plastic ($w = 0$) torque responses.

The dynamic state variable x which captures the hysteresis map, is described by the differential equation:

$$\dot{x}(t) = \Delta\dot{\theta} - \beta|\Delta\dot{\theta}||x(t)|^{n-1}x(t) - \gamma\Delta\dot{\theta}|x(t)|^n \quad (3.30)$$

The amplitude and shape of the hysteresis is controlled by the parameters β and γ . When both hysteresis parameters are zero no hysteresis loop occurs after the input direction change, the power factor $n \geq 1$ assigns the smoothness of transitions between elastic and hysteretic parts.

This equation has been implemented in Simulink for the simulations.

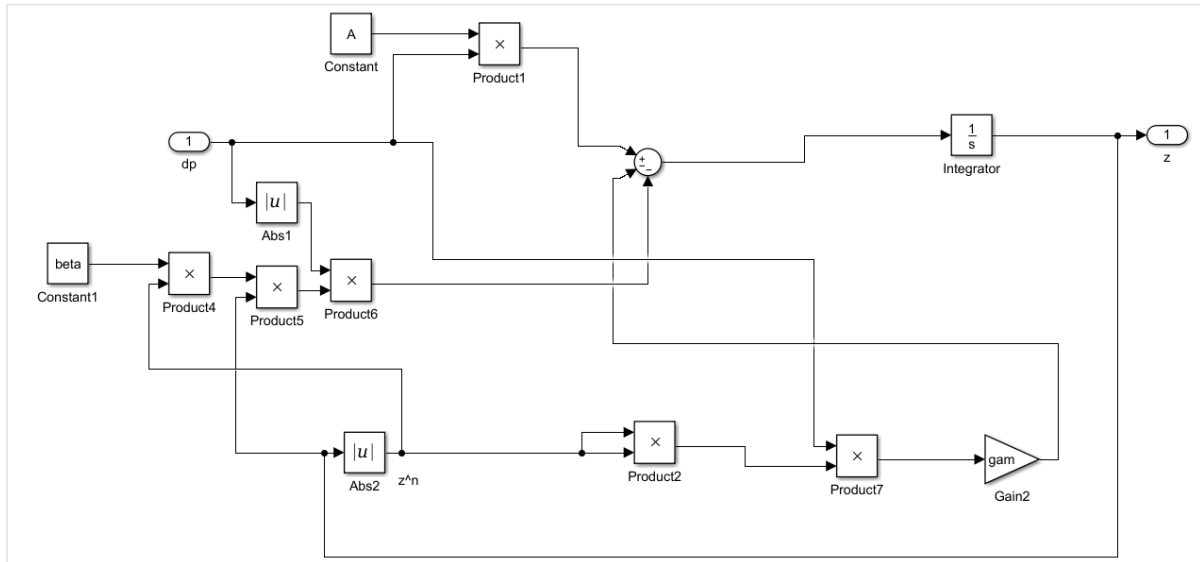


Figure 46. Internal state of the hysteretic differential equation.

This model has been studied in depth in many research as [3], [8], [12] and [15].

3.3.2 Non-linear friction modelling

Under the increasing demand of more accurate high-performance motion systems, many friction models appropriate for control purpose of mechanical systems have been proposed. These friction models formulate a dynamical model based on typical friction properties, in this project the friction studied models are the LuGre, Maxwell-Slip and Modified Maxwell-Slip, which can be found in more detail in [2], [14] and [18] for example.

LuGre friction model

The LuGre friction model was developed at the universities of Lund and Grenoble, it has arbitrary steady-state characteristics such as the Striebeck curve $g(v)$. The interpretation of the internal state is that of the bristle model, where friction is visualized as forces produced by bending bristles behaving like springs. Instead of modelling the random behaviour of friction, it is based on the average behaviour of the bristles. The average deflection of the bristles is denoted as the state variable z as shown in the following equation:

$$\frac{dz}{dt} = v - \frac{\sigma_0}{g(v)} z |v| \quad (3.31)$$

Where $g(v)$ is the Striebeck curve, which is a decreasing function for increasing velocity bounded by an upper limit equal to the static force F_s and a lower limit equal to the Coulomb force F_c .

Modelling, identification and control of an elastic joint.

$$g(v) = Fc + (Fs - Fc)e^{-(v/vs)^2} \quad (3.32)$$

The friction force is given as a function of the state variable z and the velocity v .

$$F = \sigma_0 z + \sigma_1 \frac{dz}{dt} + \sigma_2 v \quad (3.33)$$

Where the parameters $\sigma_0, \sigma_1, \sigma_2$ equal the asperity stiffness, the micro-viscous friction coefficient and the viscous friction coefficient, which as stated in the previous point is $\sigma_2 = B_m$.

Another equivalent form of the equation can be:

$$J\dot{v} = u - \sigma_0 z + \sigma_1 \frac{dz}{dt} + \sigma_2 v \quad (3.34)$$

Equations (3.31), (3.32) and (3.33) also have been implemented in Simulink.

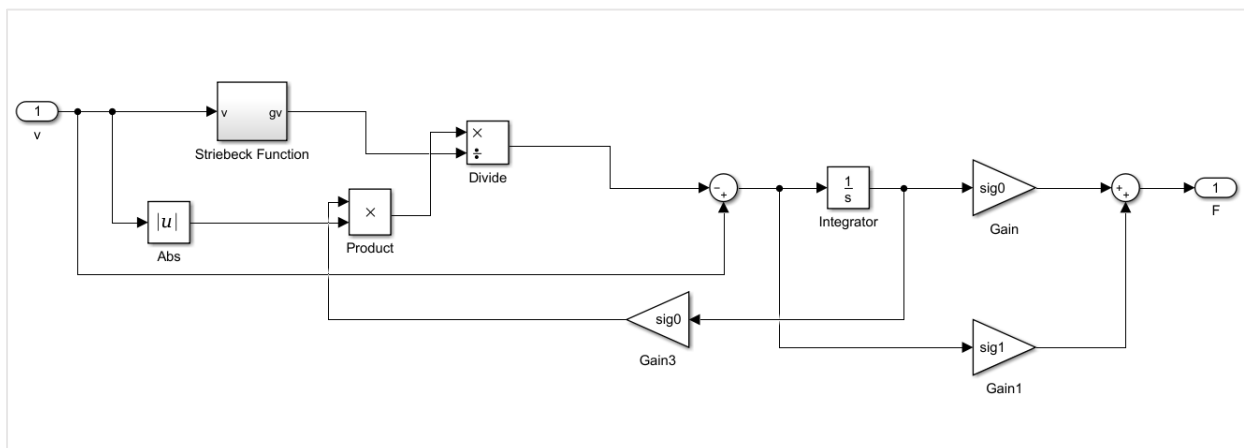


Figure 47. LuGre Simulink implementation.

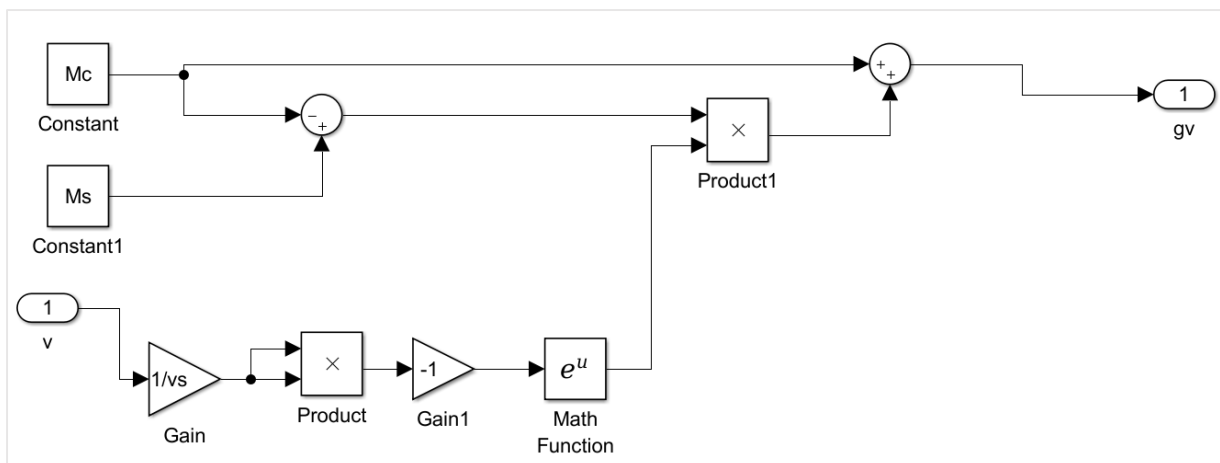


Figure 48. Striebeck function Simulink implementation.

Modelling, identification and control of an elastic joint.

In the following figure the whole system Simulink block diagram is shown.

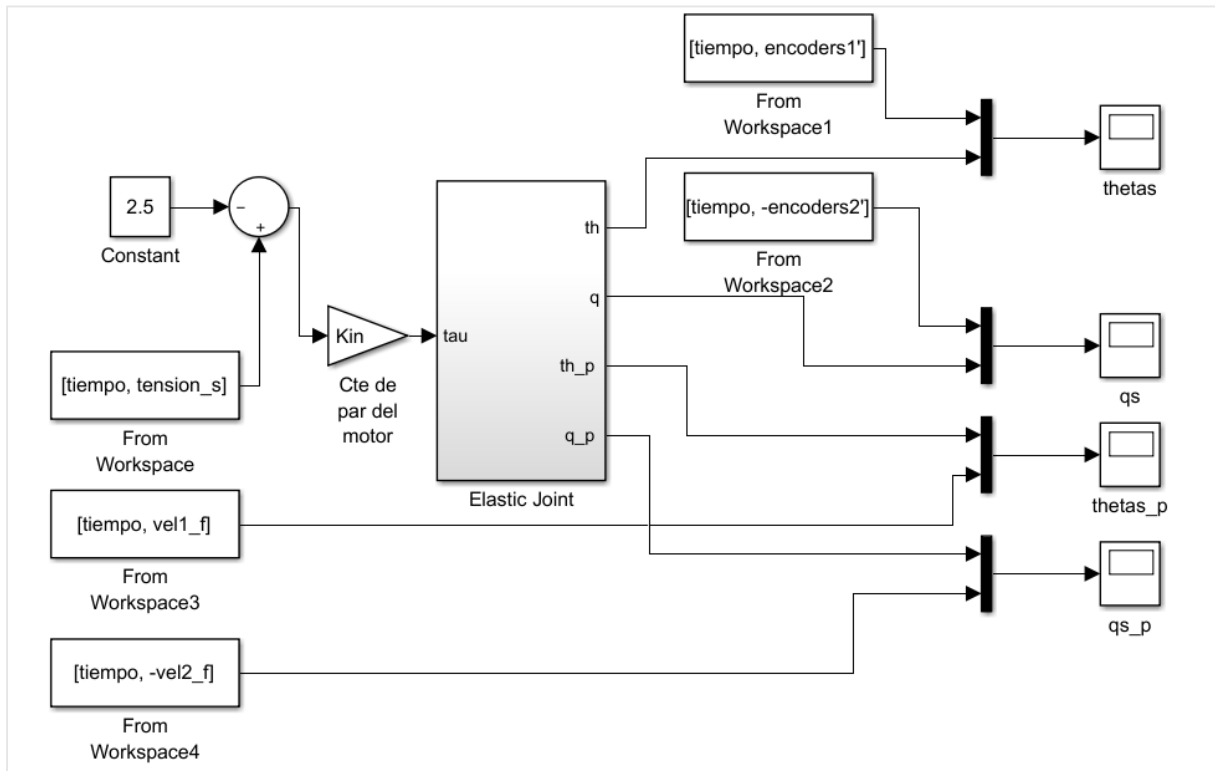


Figure 49. General view of the elastic joint implemented in Simulink.

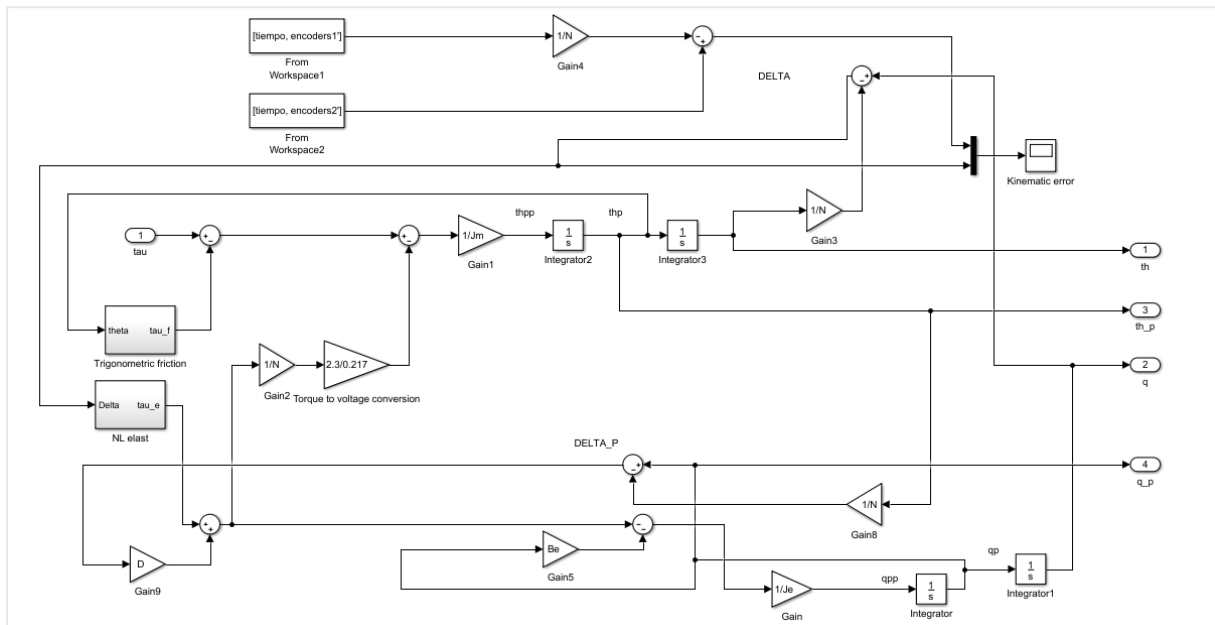


Figure 50. One joint with non-linearities Simulink representation.

Maxwell-Slip friction modelling

After the LuGre model, the design of a new model structure was based on the LuGre equations (4.31) and (4.33) that need to be adapted to describe frictional lag.

The Generalized Maxwell Slip model is a parallel connection of different single state friction, the friction force is given as the sum of the N elementary parts as seen in eq (3.35).

$$F_f(t) = \sum_{i=1}^N F_i(t) + \sigma_2 v(t) \quad (3.35)$$

So the dynamic behaviors of each elementary model can be written as:

If the elementary block is sticking:

$$\frac{dF_i}{dt} = k_i v \quad (3.36)$$

Each elementary block will remain sticking until the threshold of the breakaway force $F_i > \alpha_i s(v)$ is trespassed.

If the elementary block breaks that condition, switches its state to slipping and the differential equation is given by:

$$\frac{dF_i}{dt} = \text{sgn}(v) C(\alpha_i - \frac{F_i}{s(v)}) \quad (3.37)$$

Once the state is slipping, each elementary block resets its state to stick when the velocity crosses zero.

3.3.3 Backlash

Backlash in some gear transmissions is introduced to let the gears mesh without binding and to provide space for a film of lubricating oil between the teeth, this prevents overheating and tooth damage.

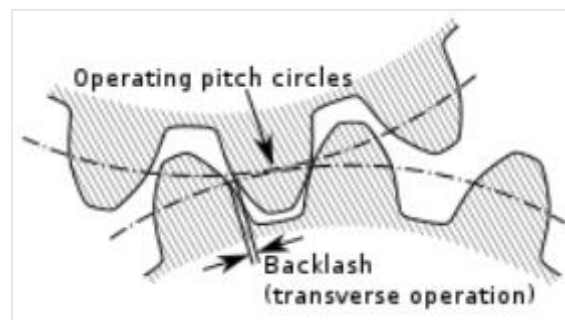


Figure 51. Backlash in the gear transmission contact teeth.

On the other hand, this causes lost motion between the reducer input and output shafts, making it difficult to achieve accurate positioning in equipment.

The mechanical gear transmission can be considered as a passive transducer of the actuator motion to the output torque that drives the joint load. The angular position of the link side constitutes the

feedback value, which contains the signature of elasticities and backlash acting in the transmission system. The gear transmissions are the main source of non-linearities as presented above.

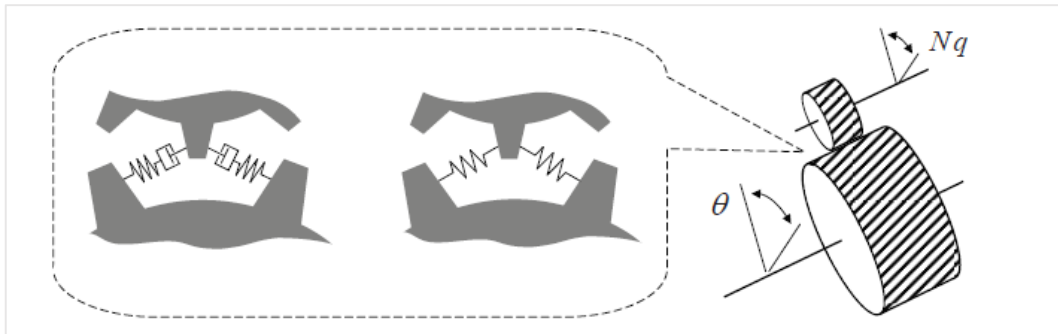


Figure 52. Gear transmission with elasticities and backlash.

Depending on the type of gear used different level of torsional compliance and backlash can be found, Harmonic Drive manufacturers state that for this kind of transmission the backlash is near zero, so this won't be included in the final model, and also can be neglected when the play size stays below the resolution of q and θ measurements.

4. System limitations

As long as this project has been developed in a low cost experimental platform, there are some limitations arising when accuracy is one of the main things taken into account when doing a research work and even more when modelling systems that will be controlled later on. Many of the researches that identify the complex physical behaviour that the Harmonic Drive transmission produce in robot joints were done by M.Ruderman, Dhaouadi and F.Ghorbel in [3], [5], [6]. These, used very sophisticated and precise equipment with encoders with higher resolution than the ones used in this project as well as torque sensors at the input and output of the gear transmission. Some others also include accelerometers to get the acceleration measurement without estimating it from the encoder position readings.

Despite this, some of the problems were solved or just considered negligible, while some others limited the goals achieved for this project, anyway all these things have been taken into consideration in order to improve or change some of the equipment used in the present project for future works.

4.3.1 Encoder saturation

When configuring the DAC pins for reading the encoder information, a timer is assigned to the pin. This timer will be “read” later on in the code and it will record the pulses read by the header. The timer has the so called auto reload, which means when it counts up to the maximum value (Depending if its 16bits or 32bits), the counting will start from zero, this effect can be appreciated in the encoder measurements for an experiment, as shown in the figure:

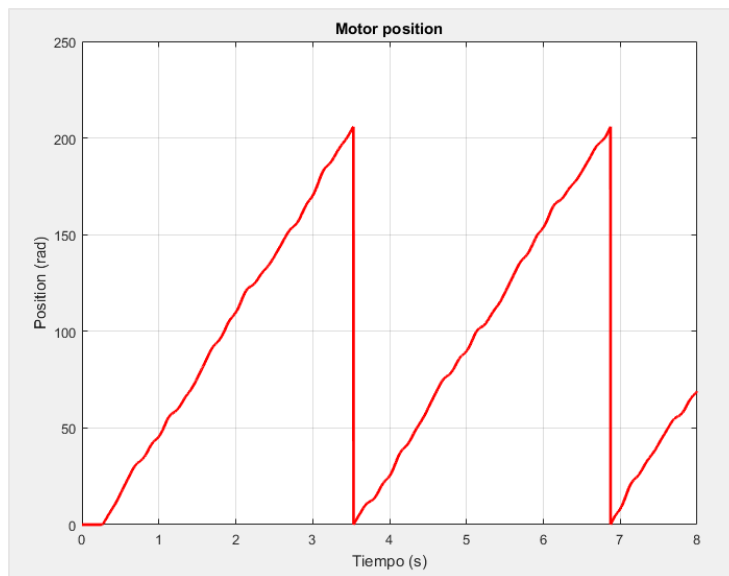


Figure 53. Encoder readings with timer auto reload.

In order to use the data in the identification procedure, the vector containing the signal has to be rearranged, for this purpose a matlab function has been created where the “jumps” are detected from the velocity estimation with the following formula:

$$v_k = \frac{\theta_k - \theta_{k-1}}{T_s} \quad (4.1)$$

So when subtracting the actual value that has “reset”, the value of v_k will be high, so this instant is used to add an offset corresponding to the “jump” value, the function implemented is shown in the following code.

```
function
[datos_ordenados]=saturacionencoder(datos_medidos,velocidad,offset,
vumbral)

N = size(datos_medidos);
saturador=0;

for i=1:N
    if(velocidad(i)<vumbral)
        saturador=saturador+offset;
    elseif(velocidad(i)>vumbral)
```

Therefore, the function takes as arguments the vector with the data with auto reload and the velocity estimated from this position to detect the auto reload instant. Then an offset equal to the maximum value that the encoder can read is added or subtracted depending on the direction of movement, the last argument is a value that is bit higher than the maximum speed reached in order to detect the high value of the estimated velocity, to know which instant has the offset to be added.

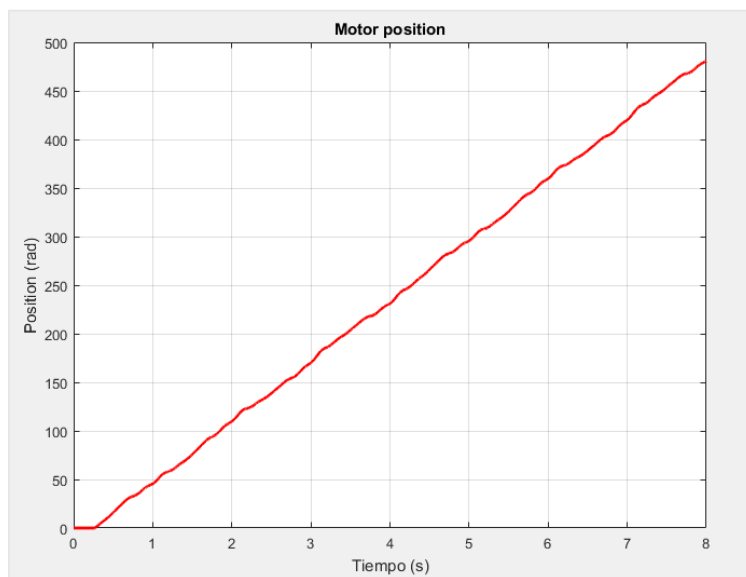


Figure 54. Encoder position fixed.

4.3.2 Velocity estimation

As explained above, the velocity is estimated from the position read by the encoders, but if the encoder has the auto reload problem, then the velocity estimated has “peaks” in the instant the auto reload happens. In order to avoid this to use the velocity as a feedback in the future if the control is implemented in the Discovery board, and to have the velocity signal already well estimated in-line without extra off-line work, the following part of the code implemented in Keil uVision fixes this problem.

```
// if timer saturates, the speed equals the previous speed .  
if (Encoder1_rad - Encoder1_rad_previous >= 200 | Encoder1_rad_previous - Encoder1_rad  
>=200) { // encoder 1  
vel_Encoder1 = vel_Encoder1_anterior;}  
else vel_Encoder1 = (Encoder1_rad - Encoder1_rad_previous) / tsampling;    // (rad/s)  
  
if (Encoder2_rad - Encoder2_rad_previous >= 200 | Encoder2_rad_previous - Encoder2_rad  
>=200) { // encoder 2  
vel_Encoder2 = vel_Encoder2_anterior;}  
else vel_Encoder2 = (Encoder2_rad - Encoder2_rad_previous) / tsampling;    // (rad/s)
```

This provides a good velocity estimation signal for the motor side, because its moving at medium speeds, this have been checked comparing the velocity estimated to the one read by the ESCON data recorder.

Nevertheless, when moving at lower speeds, this algorithm results in a bad signal estimation, as it can be seen in the following figure:

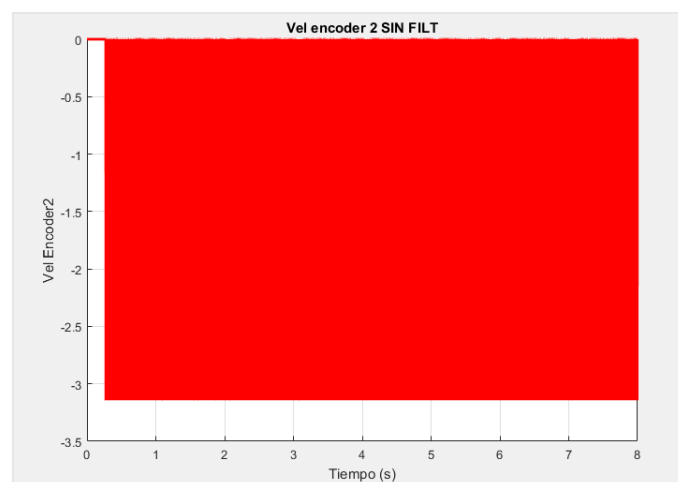


Figure 55. "Pulses" from zero to 3.14 estimated when moving at low speeds.

The main reason of this is the low resolution of the encoder (low for slow speeds) that causes the so-called quantization error as can be seen in the figure:

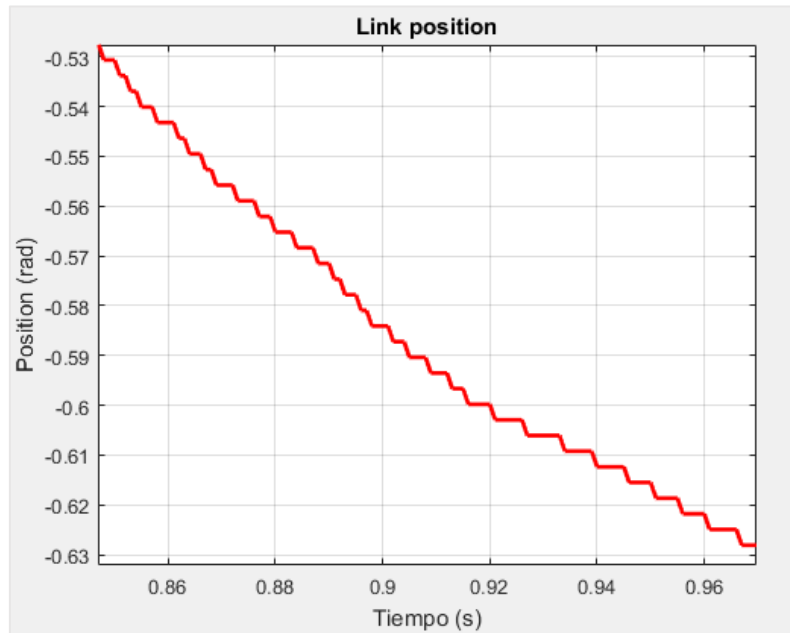


Figure 56. Quantization error in the encoder two measurements.

When calculating the speed as in equation (4.1) when the encoder value doesn't change the velocity is equal to zero, but when detecting a change the value read from the counter is converted to radians, this results in the pulses having values multiple of pi. If the velocity 2 is filtered with a no phase filter and a median filter, the result is the following.

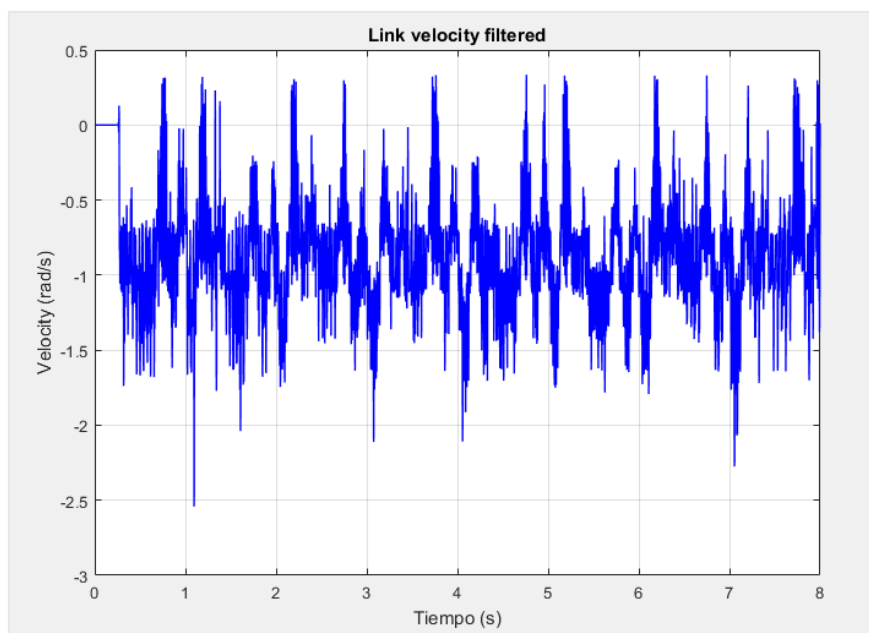


Figure 57. Velocity 2 filtered with no phase filter "filtfilt".

After the no phase filter, a medfilt can be applied to see the waveforms to have a medium value, this filter removes all the oscillations so the signal gets damped, this means that if its used to identify the elasticity probably a lot of information will be lost when applying the filter and the results obtained are bad.

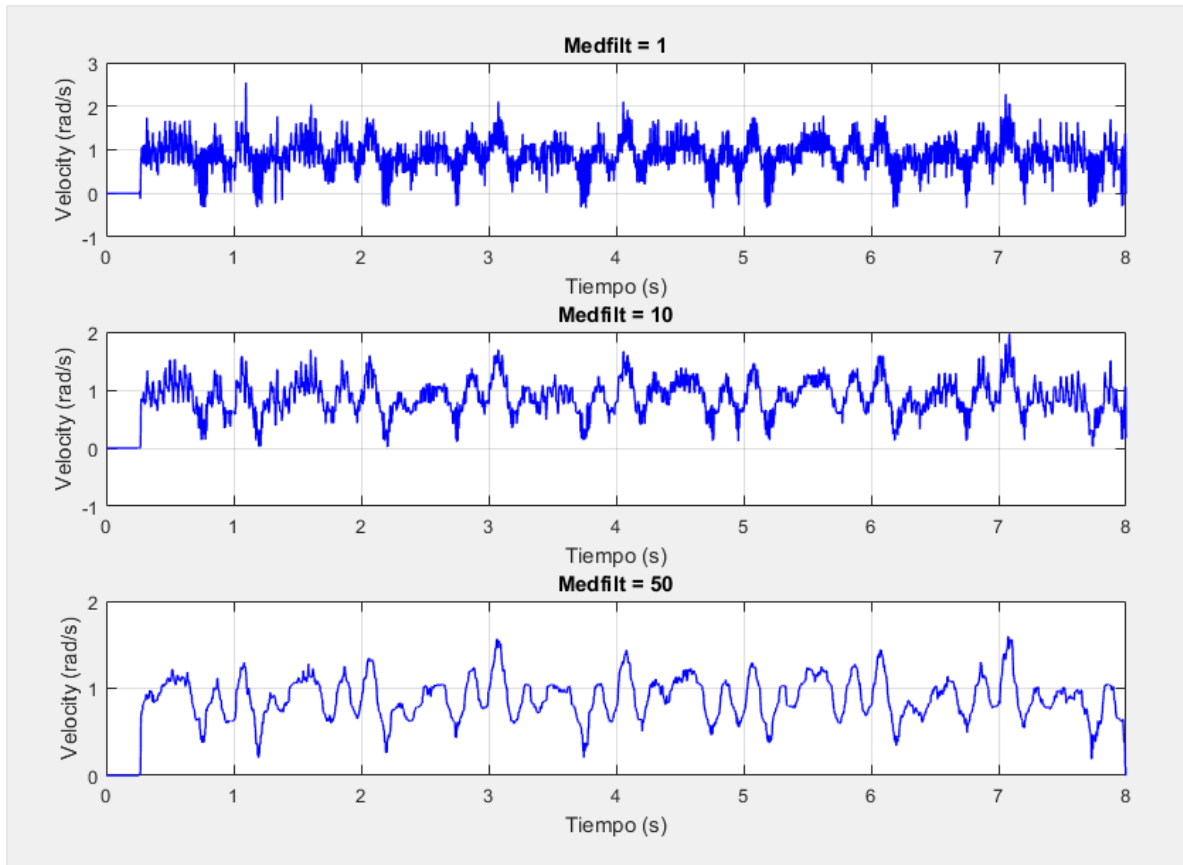


Figure 58. Velocity 2 after medfilt with values 1, 10 and 50.

4.3.3 DAC mismatch

In the identification procedure, the input signal has to be chosen depending on which physical phenomena is going to be analysed. When programming the microcontroller, the waveform defined as a function of the time is recorded in the SDRAM and then read as a value as if it were another measurement. So when reading this value, it has to be taken into account that it's the value which has been demanded to the DAC and this can differ from the real value the DAC is sending, this can be checked when comparing the DAC signal read from the SDRAM and the signal measured in the ESCON and recorded in the data analyzer from the ESCON Studio.

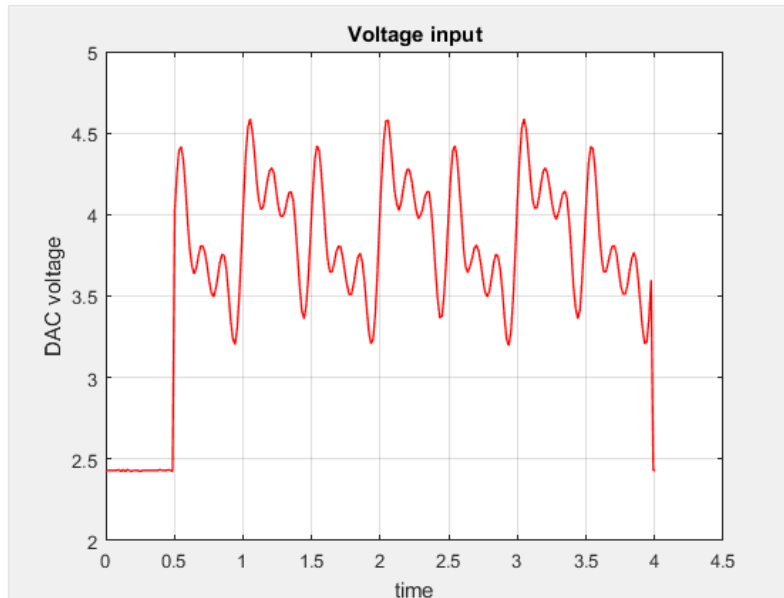


Figure 59. ESCON analog input2.

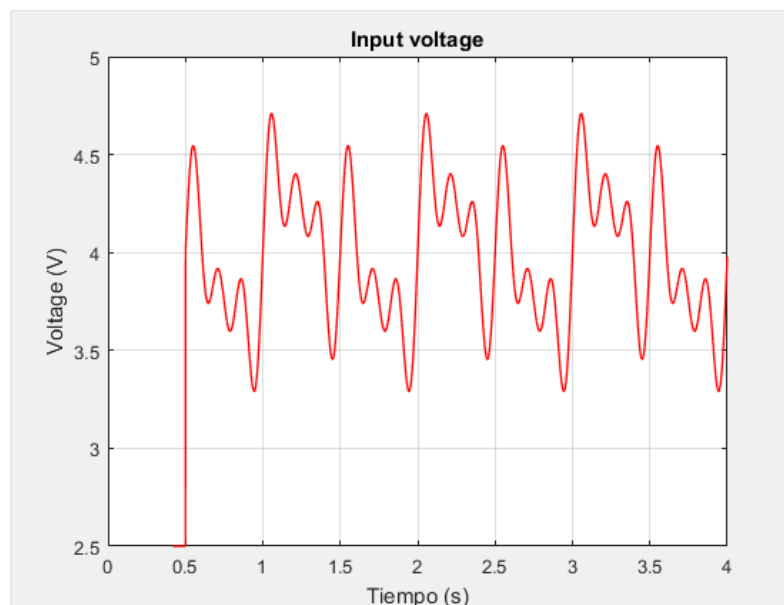


Figure 60. DISCOVERY input voltage data.

When comparing the peaks of the signals from the DAC at the instant 0.6 seconds the value is above 4.5 Volts while the signal measured in the ESCON at the same instant the value doesn't reach the 4.5 Volts, this mismatch can be appreciated all along the signal. The ideal case would be to use the ESCON measurement, which in fact is more accurate, but as long as this error is small, the data read from the SDRAM will be used, even though some trials and comparisons will be made using the ESCON measurements.

4.3.4 Speed saturation in current control

When operating in current control, the DAC signal that is Analog input to the ESCON is defined as a mathematical equation in the Keil code. The current control allows the current sent to the motor to be equal to the signal generated by the DAC, with this usual configuration used in robotics, the torque applied by the motor is estimated from the torque constant of the motor K_a , which is proportional to the current as related in the following formula.

$$\tau_m(t) = K_a i(t) \quad (4.2)$$

In the case of BLDC, three phase motor, the formula as stated in [26] if the three phases are equal:

$$\tau_m(t) = 2K_a i(t) \quad (4.3)$$

Where the current is measured in one of the phases.

Therefore, when configuring the ESCON in current control the relationship between the DAC value and the current requested has to be set according to the specifications of the system, in this case the current nominal for the motor is 1.78A and the maximum voltage the DAC can send is 5V so knowing this the voltage to current constant given by:

$$Kiv = \frac{1.78 \text{ A}}{5 \text{ V}} = 0.356 \text{ A/V} \quad (4.4)$$

Therefore, the set value of the current for the control loop in the ESCON studio is given by:

$$i_{set_value} = 0.356 \cdot V_{in} \quad (4.5)$$

Once we know the operating point in which the motor is working, the boundaries to the input signal are calculated, this is because there is speed saturation for the upper boundary input voltage and a friction force to break for the lower boundary input voltage. In addition, the minimum input voltage that breaks the friction in steady state will be estimated.

Now the main goal is to determine which are the boundaries of the input value that makes no saturation in the speed, in order to set sinusoidal waves inside these values to get velocity measurements with no saturation.

The first experiment will determine the minimum value that breaks the friction for the given operating point, this is achieved increasing the value of the step at the input, until the minimum value for breaking the friction is found to be $V_{min_friction_break} = 0.9 \text{ V}$.

The next experiments will determine the minimum V_{in} that keeps the velocity constant in steady state.

The input signal will have an initial peak with 0.9V for breaking the friction and then will set into a constant value.

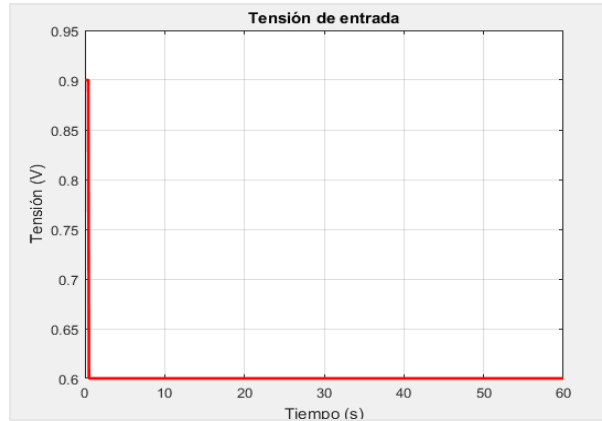


Figure 61. Input with initial peak for friction break.

The following table shows the properties from the velocity estimated at the given V_{in} for $N = 20000$, this means 20s experiments with 0.001s as sample time.

Table 10. Motor response in current control mode.

V_{in} (V)	t_s (s)	Speed sat	Friction break	Stop SS	Input Signal (V)
0.4	4.5	No	Initial peak	No	0.9 / 0.4
0.3	-	-	Initial peak	Yes	0.9 / 0.3
0.35	-	-	Initial peak	Yes	0.9 / 0.35
0.375	-	-	Initial peak	Yes	0.9 / 0.375

From the table above it can be determined that the minimum V_{in} to move in steady state without stopping because the friction force is $V_{min_ss} = 0.4 V$.

Last value to be determined is the minimum voltage input in steady state that does not reach the speed saturation zone.

Since it is not known if the minimum value for breaking the friction ($V_{min_friction_break} = 0.9 V$) makes the speed saturate, the first approach will be using the following input signal.

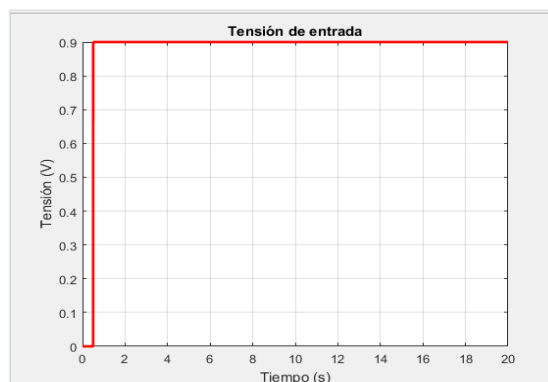


Figure 62. Input signal, step at 0.9V.

From this input signal the following results are given:

Table 11. Motor response in current control mode.

V_{in} (V)	t_s (s)	Speed sat	Friction break	Stop SS	Input Signal (V)
0.9	2	Yes	Yes	-	0 / 0.9
0.8	-	-	No	-	0 / 0.8

Therefore, in the given context we cannot break friction with a constant value without saturating the output speed.

Then the input signal has to be changed since the maximum value at the input voltage has to be known for defining the sinusoidal input in the following experiments, this means the initial peak at 0.9V and then constant value will be used again as input signal.

Table 12. Motor response in current control mode.

V_{in} (V)	t_s (s)	Speed sat	Friction break	Stop SS	Input Signal (V)
0.7	6	Yes	Initial peak	x	0.9 / 0.7
0.6	Curve +7	No	Initial peak	x	0.9 / 0.6
0.65	Curve +7	Yes, at some points	Initial peak	x	0.9 / 0.65
0.625	Curve +7	No	Initial peak	x	0.9 / 0.625 N=30k
0.637	Curve +7	No	Initial peak	x	0.9 / 0.637 N=20k

From the last experiment, it can be concluded that the maximum input voltage that does not saturate speed at steady state is $V_{max_ss} \approx 0.64 V$.

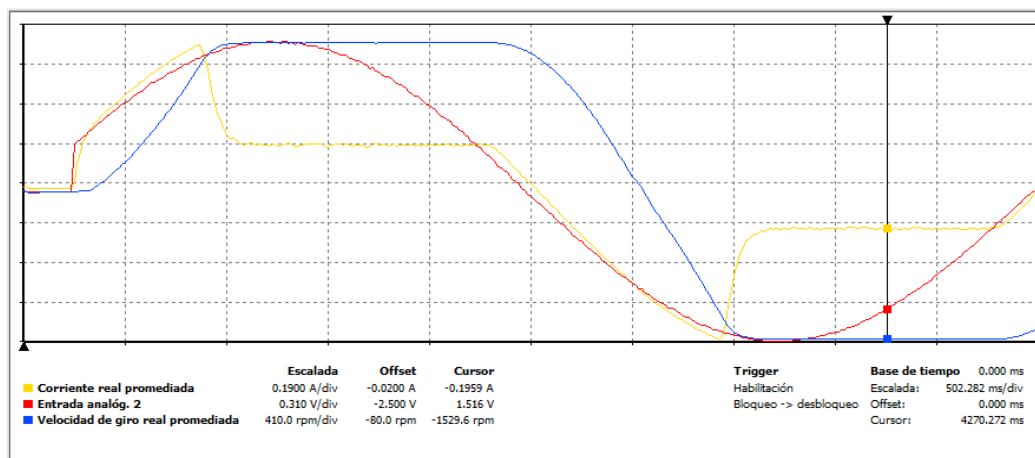


Figure 63. ESCON data record showing speed (voltage) saturation and current (torque) saturation when reached the maximum speed.

4.3.5 ESCON data recorder low sampling time

The ESCON Studio software has a tool to monitorize the signals of interest as the current in the phase, the voltage drop, the analog signal received from the Discovery DAC. The problem using this data for identification tasks is that it has limited data storage with a maximum of N samples and the user cannot modify this value. Instead, the time of the experiment is set and the sample period is adjusted in order to fit the data spaced with the same interval of time, in the figure below an example of this data record is shown.

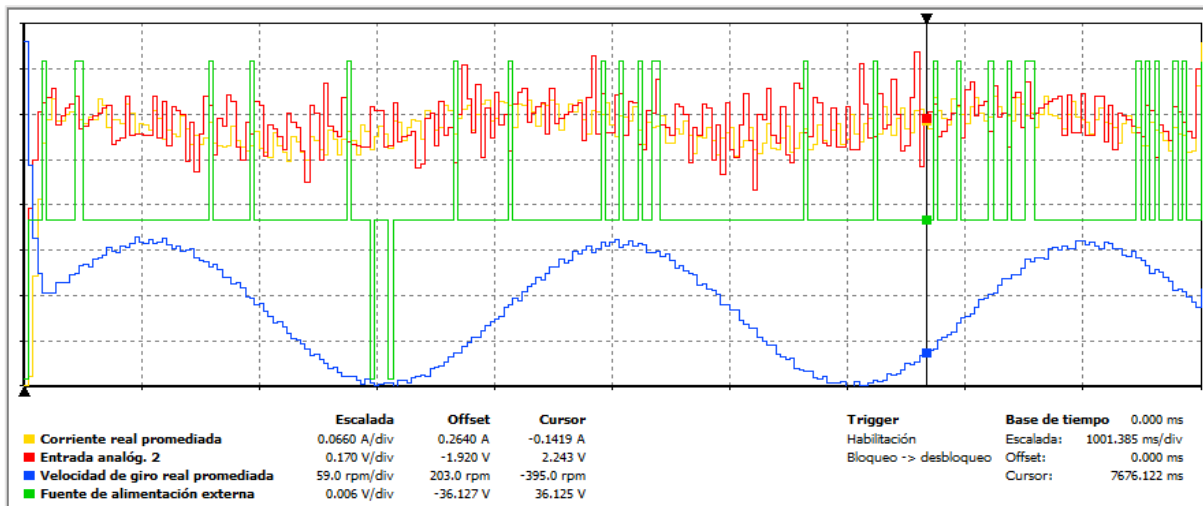


Figure 64. ESCON Studio data recorder.

The sample time of the ESCON depends on how long is the experiment, this means that it has a fixed number of samples (256) so depending on the time of the experiment the sampling time is calculated to fit the 256 samples in the given time.

Therefore, there is a mismatch between the sampling time of the Discovery measurements and the ESCON measurements. In Matlab, this data can be treated with an interpolation script where a time series object is created containing the data and its corresponding time vector with different sample times. Then, using the command `resample`, the data is interpolated in order to obtain a vector with the same sample time using a linear hold. So if the experiment is set to be short, the sample time difference will be smaller so there will be less error in the measurements by using this method, the script implemented in Matlab is shown below.

```
% Transform the data to the same sample time
t_escon = linspace(0,9.028,256);
ts_escon = timeseries(VarName2,t_escon);
ts_out = resample(ts_escon, tiempo);
current = ts_out.Data;
```


In this example, the experiment length is 9.028 seconds, this value is calculated by the ESCON in order to fit the number of samples spaced with the same sample time, then the time series created is resampled giving as argument the new time vector with the Discovery sampling time, then the data is extracted from the time series data.

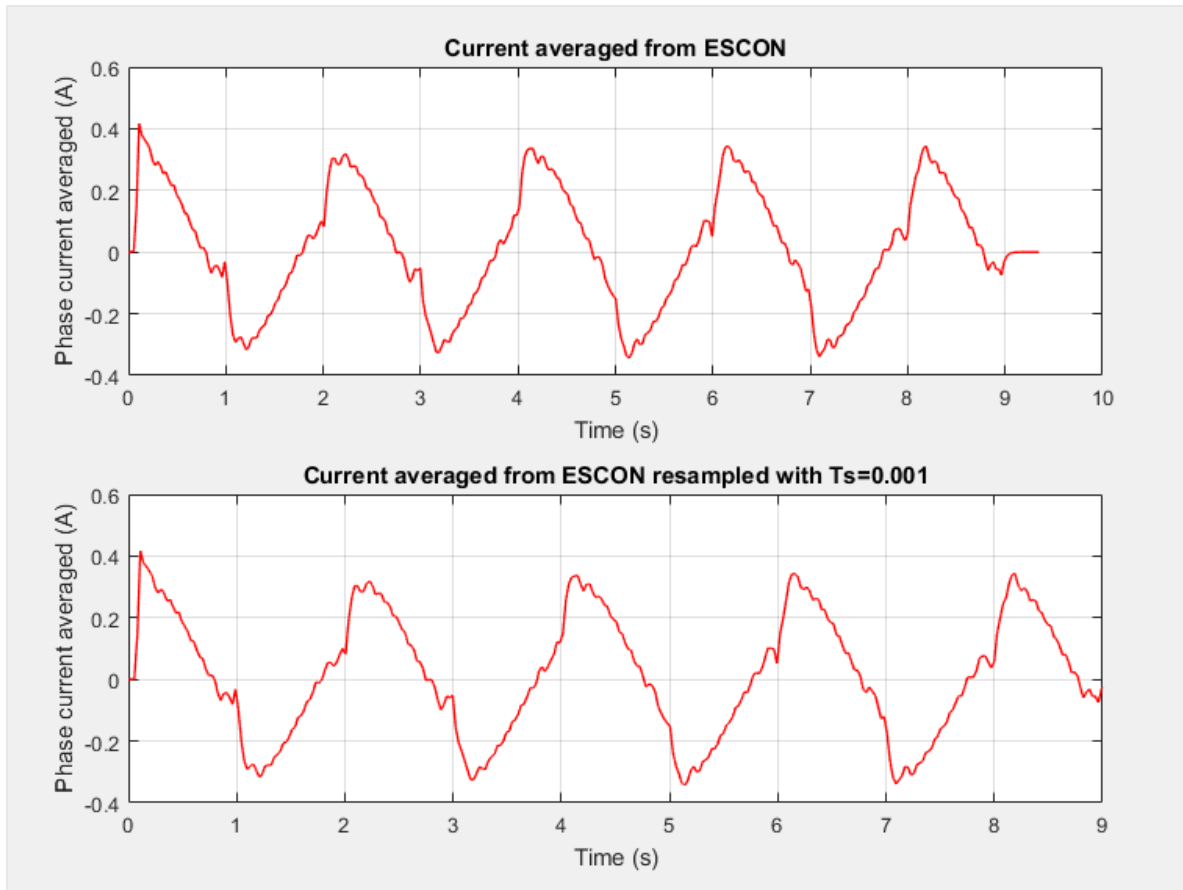


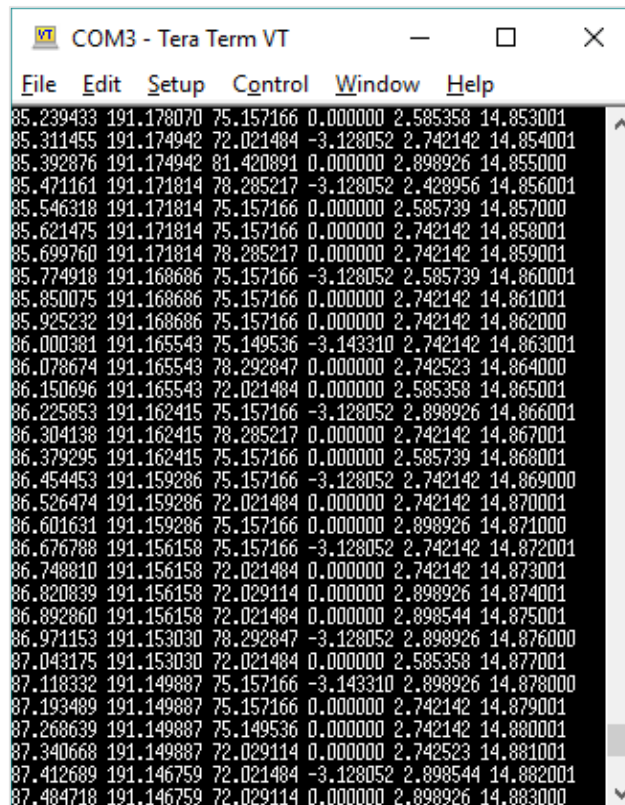
Figure 65. Current acquired from ESCON and resampled signal.

5. Identification

The construction of a model from data involves three basic entities:

1. A data set, with N samples
2. A set of candidate models
3. A rule by which candidate models can be assessed using the data, like the Least Squares selection rule.

The input-output data are sometimes recorded during a specifically designed identification experiment, where the used may determine which signals to measure and when to measure them and may choose the input signals. The objective with experiment design is thus to make these choices so that the data become maximally informative, subject to constraints that may be at hand.



```
COM3 - Tera Term VT
File Edit Setup Control Window Help
85.239433 191.178070 75.157166 0.000000 2.585358 14.853001
85.311455 191.174942 72.021484 -3.128052 2.742142 14.854001
85.392876 191.174942 81.420891 0.000000 2.898926 14.855000
85.471161 191.171814 78.285217 -3.128052 2.428956 14.856001
85.546318 191.171814 75.157166 0.000000 2.585739 14.857000
85.621475 191.171814 75.157166 0.000000 2.742142 14.858001
85.699760 191.171814 78.285217 0.000000 2.742142 14.859001
85.774918 191.168686 75.157166 -3.128052 2.585739 14.860001
85.850075 191.168686 75.157166 0.000000 2.742142 14.861001
85.925232 191.168686 75.157166 0.000000 2.742142 14.862000
86.000381 191.165543 75.149536 -3.143310 2.742142 14.863001
86.078674 191.165543 78.292847 0.000000 2.742523 14.864000
86.150696 191.165543 72.021484 0.000000 2.585358 14.865001
86.225853 191.162415 75.157166 -3.128052 2.898926 14.866001
86.304138 191.162415 78.285217 0.000000 2.742142 14.867001
86.379295 191.162415 75.157166 0.000000 2.585739 14.868001
86.454453 191.159286 75.157166 -3.128052 2.742142 14.869000
86.526474 191.159286 72.021484 0.000000 2.742142 14.870001
86.601631 191.159286 75.157166 0.000000 2.898926 14.871000
86.676788 191.156158 75.157166 -3.128052 2.742142 14.872001
86.748810 191.156158 72.021484 0.000000 2.742142 14.873001
86.820839 191.156158 72.029114 0.000000 2.898926 14.874001
86.892860 191.156158 72.021484 0.000000 2.898544 14.875001
86.971153 191.153030 78.292847 -3.128052 2.898926 14.876000
87.043175 191.153030 72.021484 0.000000 2.585358 14.877001
87.118332 191.149887 75.157166 -3.143310 2.898926 14.878000
87.193489 191.149887 75.157166 0.000000 2.742142 14.879001
87.268639 191.149887 75.149536 0.000000 2.742142 14.880001
87.340668 191.149887 72.029114 0.000000 2.742523 14.881001
87.412689 191.146759 72.021484 -3.128052 2.898544 14.882001
87.484718 191.146759 72.029114 0.000000 2.898926 14.883000
```

Figure 66. Terminal receiving data example.

Before explaining in detail all the procedures and steps done in the project a flux diagram is presented where all the information about the identification process is shown, this step by step process can be applied to all the identifications done in the present project.

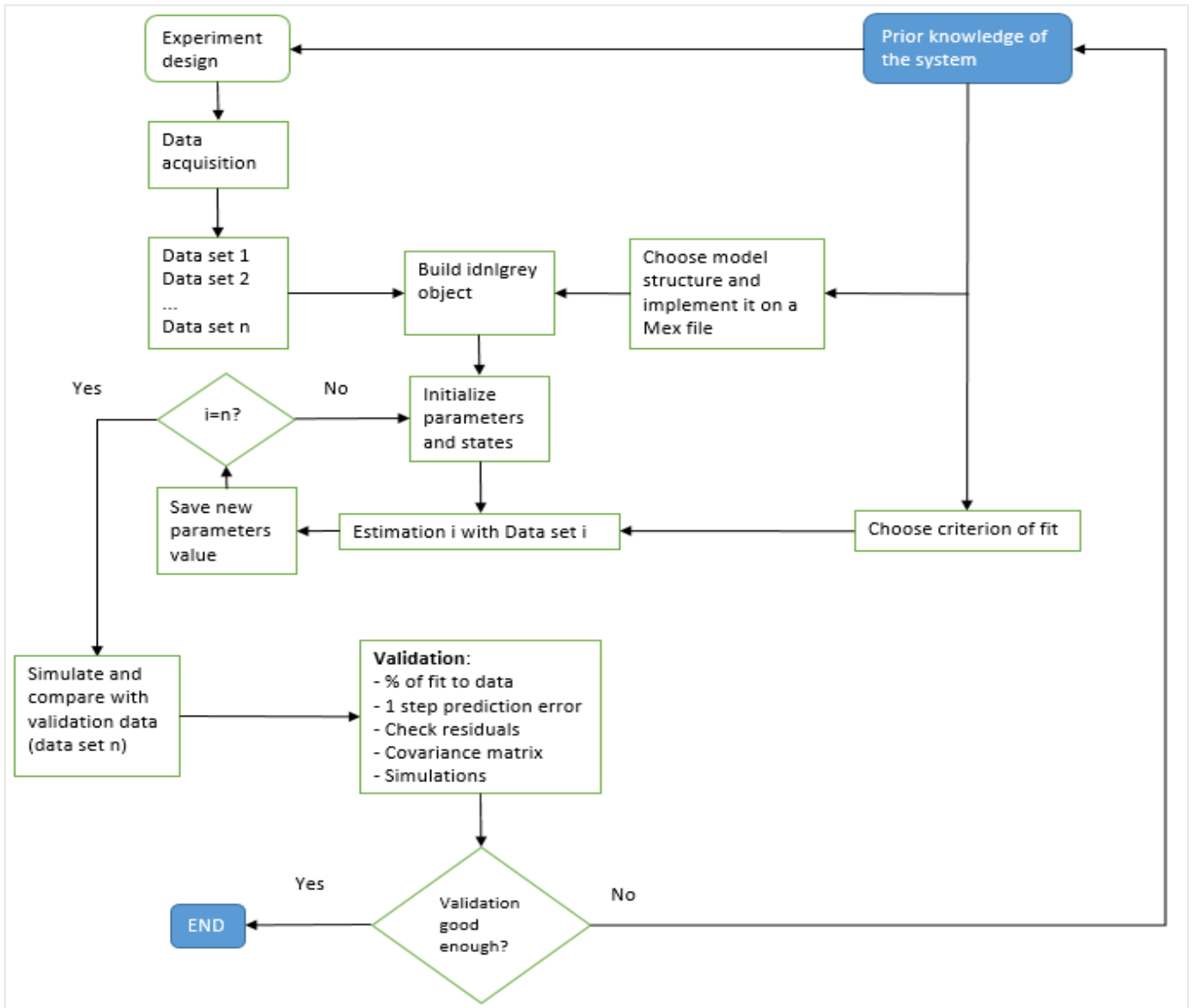


Figure 67. Flux diagram of the identification procedure.

Now, the procedure used for taking the data from the experiments is explained in detail.

1. Initialization of the system

First, turning on the ESCON supply, once the green LED starts blinking, we can connect the ESCON to the computer by USB, it is important to do it in this order, otherwise the supply has to be disconnected and connected again to connect properly the ESCON to the computer. The Discovery is supplied when connected to the PC via USB, this is also the way for loading the code to the board via the ST-LINK driver. The mini USB for data transmission is also connected to the computer, the code has been designed in a way that this connection can be made once the experiment has finished, and this means the Discovery is not sending data until the mini USB connection is done.

2. ESCON configuration

Before loading the code and executing, the ESCON configuration will be checked, also the data recorder will be prepared, the ESCON studio software provides a way of triggering the data record when enabled with the GPIO logic signal sent, also there are four channels where the user can choose which signals are monitored.

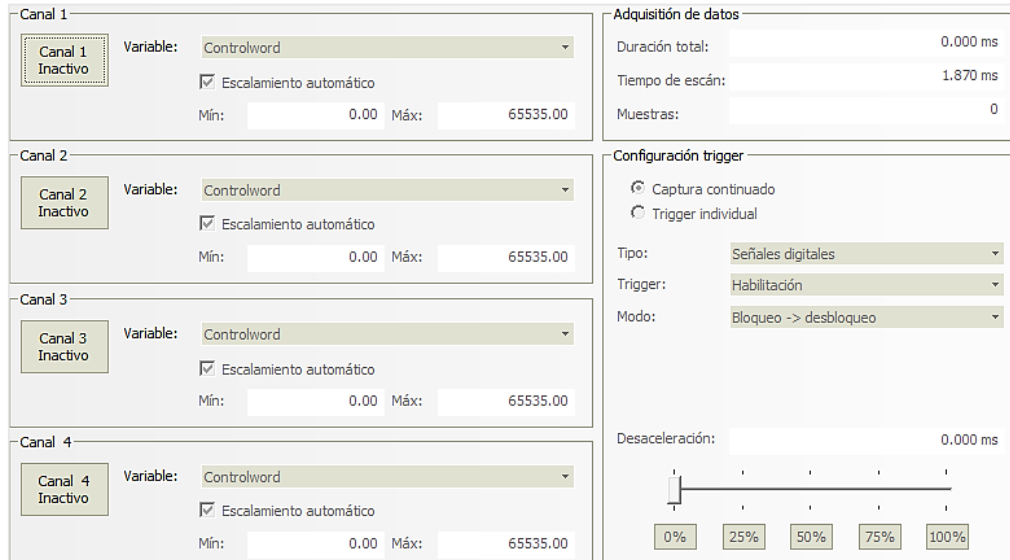


Figure 68. ESCON studio data recorder configuration.

So depending on the experiment that will be done, the voltage or current control is selected, then the offsets and other control parameters.

3. Keil uVision code load

Once the ESCON is prepared the code, containing the identification experiments is loaded to the board. When the program is loading, some of the functionalities in the Discovery are reset, so the user has to be careful because the DAC will reach its maximum value for some seconds, if the ESCON is enabled the motor may run at maximum speed for a few seconds, this is anyway solved with the enable/disable input in the ESCON. If some changes are made in the code, the project has to be built and loaded again.



Figure 69. Compile and load buttons on Keil uVision.

After this, the experiment will run and the data will be recorded into the SDRAM.

```
TM_SDRAM_WriteFloat(aux, Encoder1_rad + saturador );  
TM_SDRAM_WriteFloat(aux+8, Encoder2_rad + saturador2 );  
TM_SDRAM_WriteFloat(aux+16, vel_Encoder1);  
TM_SDRAM_WriteFloat(aux+24, vel_Encoder2);  
TM_SDRAM_WriteFloat(aux+32, tension_value);  
TM_SDRAM_WriteFloat(aux+40, tiempo );  
aux += 48;
```

Figure 70. Code writing data on the SDRAM.

4. Run the experiment

The experiment will be running for the time specified in the code, when the experiment ends the data stored in the SDRAM waits for the VCP USB connection to start sending data and receive it on the personal computer.

```
if (TM_USB_VCP_GetStatus() == TM_USB_VCP_CONNECTED) {
```

Figure 71. Data is not transmitted until the state is true.

Anyway, as long as the Teraterm has to be turned on and the serial port where the USB is connected has to be selected, it is better to let the USB connected from the beginning of the experiment.

5. Receive data in Teraterm

When opening the Teraterm, the following window is shown:

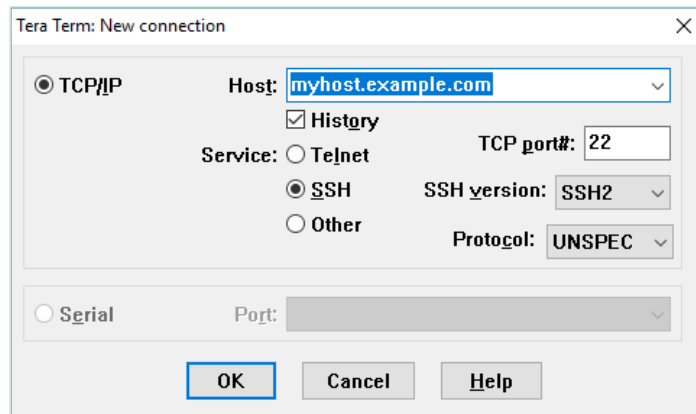


Figure 72. Teraterm connection options.

The option chosen is the serial port, the one that has the STMmicroelectronics name, after that the data will be received, but the file has to be saved with extension “.m” to open it with Matlab.

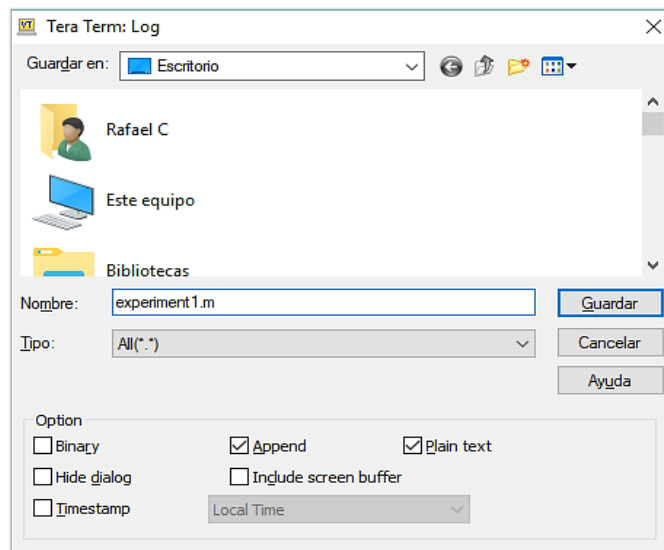


Figure 73. Log save screen.

6. Load the data to Matlab

After finishing the data transmission, the log saved can be open with Matlab, in the code the data is saved as a matrix with of size $N \times 6$ where, N , is the number of samples in the experiment, this value is calculated knowing the sampling time $T_s = 0.001s$ and the experiment duration.

$$N_{samples} = \frac{T_{experiment}}{T_s} \quad (5.1)$$

Then the following matlab script reads the matrix:

```
%% EXTRAER DEL TERATERM
Posencoder1 = A(:,1);
Posencoder2 = A(:,2);
vell_noise = A(:,3);
vel2_noise = A(:,4);
tension_s = A(:,5);
tiempo = A(:,6);
```

Figure 74. Matlab data script.

Encoder position 1 and position 2 are directly measured. The velocity is estimated by differentiation, the time is obtained from the control timer and the signal of the voltage from the value given by the math functions implemented in the code, this value might vary from the real voltage provided by the DAC, this can be seen when comparing the ESCON signal to this one.

As long as the velocities are estimated there is a need to filter the signals in order to remove noise but without affecting the phase or amplitude of the waves.

```
%% Velocity filtering
%Vel 1 Filter
[b,a] = butter(12,0.3,'low');

vell_f = filtfilt(b,a,vell_noise);
vell = medfilt1(vell_f,10);

%Vel 2 Filter
vel2_f = filtfilt(b,a,vel2_noise);
vel2 = medfilt1(vel2_f,100);
```

Figure 75. Velocity filter applied.

The kinematic error also has to be calculated from both measurements at the input and output side, with given configuration (input and output with opposite directions) the kinematic error is calculated as follows:

```
%% Delta estimation from q and th
%(IF Q ND TH SAME DIRECTION -> q-th/N) (IF Q- and TH+ -> q+th/N)

tors_angle = (encoders1'/70)+encoders2';
```

Figure 76. Kinematic error estimation.

This matlab script also contains all the plots for all the signals needed.

```
%% PLOTS
% %ENCODER1
figure
p = plot(tiempo,encoders1')
% p = plot(tiempo(3000:end),encoders1(3000:end)')
% p = plot(tiempol,encoders1)
title('Motor position');
xlabel('Tiempo (s)');
ylabel('Pos Encoder1');
set(p, 'Color', 'red', 'LineWidth', 2)
grid
```

Figure 77. Signal plot script.

Depending on the kind of experiment carried, it may be convenient to choose the range of the signal that wants to be visualized, or to know which are the initial states of the identification procedure.

Once all the data is ready, the identification procedure can start, before that, the search methods available for solving nonlinear data fitting problems with Matlab are presented.

5.1 Methods for nonlinear least squares curve-fitting problems

The data acquisition procedure has been explained; the models have been studied and implemented, now the last point is to know the procedures used in order to fit the data into the models by adjusting the parameters in the model.

Least squares problems arise in the context of fitting a parametrized function to a set of measured data points by minimizing the sum of the squares of the errors between the data points and the function. If the fit function is non-linear in the parameters, then the LS problem is nonlinear.

Nonlinear LS methods iteratively reduce the sum of the squares of the errors between the function and the measured data points through a sequence of updates to parameter values. The Levenberg-Marquardt curve fitting method is a combination of two minimization methods:

- Gradient descent method: The sum of the squared errors is reduced by updating the parameters in the steepest-descent direction.
- Gauss-Newton method: The sum of the squared errors is reduced by assuming the LS function is locally quadratic, and finding the minimum of the quadratic.

The LM method acts more like a gradient descent method when the parameters are far from their optimal value, and acts more like the Gauss-Newton method when the parameters are close to their optimal value.

Knowing this it would be appropriate to use a gradient descent method with the first estimation data set, and then apply GNA or LM method for validation purposes.

When fitting a function $\hat{y}(t, \rho)$ of independent variable t and vector of n parameters ρ , to a set of N data points (t_i, y_i) , it is customary and convenient to minimize the sum of the weighted squares of the errors (or weighted residuals) between the measured data $y(t_i)$ and the curve fit function $\hat{y}(t, \rho)$. This goodness of fit measure is called the chi-squared error criterion and is defined as follows:

$$\chi^2(\rho) = \sum_{i=1}^N \left[\frac{y(t_i) - \hat{y}(t, \rho)}{\sigma_{y_i}} \right]^2 \quad (5.2)$$

$$= (Y - \hat{Y}(\rho))^T W (Y - \hat{Y}(\rho)) \quad (5.3)$$

$$= Y^T W Y - 2Y^T W \hat{Y} + \hat{Y}^T W \hat{Y} \quad (5.4)$$

Where σ_{y_i} is the measurement error for measurement $y(t_i)$.

Typically, the weighting matrix W is diagonal with $W_{ii} = 1/\sigma_{y_i}$.

If the function is nonlinear in the model parameters then the minimization of chi square with respect to the parameters must be carried out iteratively. The goal of each iteration is to find a perturbation h to the parameters ρ that reduces χ^2 .

5.1.1 The gradient descent method

The steepest descent method is a general minimization method that updates the parameters in the direction opposite to the gradient of the objective function. This method converges well for problems with simple objective functions. For problems with many parameters gradient descent methods are sometimes the only viable choice.

Said this, the gradient of the chi-squared objective function with respect to the parameters is:

$$\frac{\partial}{\partial \rho} \chi^2 = 2 (Y - \hat{Y}(\rho))^T W \frac{\partial}{\partial \rho} (Y - \hat{Y}(\rho)) \quad (5.5)$$

$$= -2 (Y - \hat{Y}(\rho))^T W \left[\frac{\partial \hat{Y}(\rho)}{\partial \rho} \right] \quad (5.6)$$

$$= -2 (Y - \hat{Y}(\rho))^T W J \quad (5.7)$$

Where the $m \times n$ Jacobian matrix $\left[\frac{\partial \hat{Y}(\rho)}{\partial \rho} \right]$ represents the local sensitivity of the function $\hat{y}(t, \rho)$ with respect to the parameters. As mentioned before, we search for the parameter update h , this update moves the parameters value in the direction of steepest descent, and is given by:

$$h = \alpha J^T W (Y - \hat{Y}) \quad (5.8)$$

Where α is the length of the step in the steepest-descent direction.

5.1.2 Gauss-Newton method

This method minimizes the sum of squares objective function, the objective function is approximately quadratic in the parameters near the optimal solution, if the problem is not big sized, this method converges much faster than gradient descent method.

The function evaluated with perturbed model parameters may be locally approximated through a first-order Taylor series expansion.

$$\hat{y}(\rho + h) \approx \hat{y}(\rho) + \left[\frac{\partial \hat{Y}(\rho)}{\partial \rho} \right] h = \hat{y} + Jh \quad (5.9)$$

Substituting the approximation for the perturbed function, $\hat{y} + Jh$ for \hat{y} in the equation (x[3]) gives:

$$\chi^2(\rho + h) \approx Y^T W Y + \hat{Y}^T W \hat{Y} - 2Y^T W \hat{Y} - 2(Y - \hat{Y})^T W J h + h^T J^T W J h \quad (5.10)$$

This shows that χ^2 is approximately quadratic in the perturbation h , and that the Hessian of the chi-squared fit criterion is approximately $J^T W J$.

The parameter update h that minimizes χ^2 is found from $\frac{\partial}{\partial h} \chi^2 = 0$, so:

$$\frac{\partial}{\partial h} \chi^2(\rho + h) \approx -2(Y - \hat{Y})^T W J + 2h^T J^T W J \quad (5.11)$$

Moreover, the resulting normal equations for the Gauss-Newton update are

$$[J^T W J]h = J^T W (Y - \hat{Y}) \quad (5.12)$$

5.1.3 The Levenberg-Marquardt method

The Levenberg-Marquardt algorithm adaptively varies the parameter updates between the gradient descent update and the Gauss-Newton update.

$$[J^T W J + \lambda I] h_{lm} = J^T W (Y - \hat{Y}) \quad (5.13)$$

Where small values of the algorithmic parameter λ result in a Gauss-Newton update and a large value results in a gradient descent update. The parameter λ is initialized to be large so that first updates are small steps in the steepest-descent direction. If any iteration happens to result in a worse approximation $\chi^2(\rho + h_{lm}) > \chi^2(\rho)$ then λ is increased. Otherwise, as the solution improves, lambda is decreased so the LM method approaches the GN method and the solution typically accelerates to the local minimum.

In the following Marquardt's, update relationship

$$[J^T W J + \lambda \text{diag}(J^T W J)] h_{lm} = J^T W (Y - \hat{Y}) \quad (5.14)$$

The values of lambda are normalized to the values of $J^T W J$.

5.2 Reduction ratio comprobatation

First, the reduction ratio of the gear will be analysed, as the manufacturer stated, the reduction ratio is 70, this is anyway easy to check with simple experiments where the motor is moved and both measurements at the input and output are used to calculate it as follows:

$$N = \theta/q \quad (5.15)$$

The provided gear ratio is ideal, this is however hard to achieve in the real world, imperfections in the manufacturing or assembly may vary the real value specified; also this can occur by error in the measurements as long as sensors aren't perfect.

For determining the experimental value of the reduction ratio, different inputs will be used and the results between the experiments will be compared, these experiments consist on a constant step, stair input and a sum of sinusoidals. The following lines of code allows us to estimate N from the positions vectors.

```
A=encoders1(1000:end);  
B=encoders2(1000:end);  
N=-A/B
```

As long as both directions are opposite due to the H.D configuration (Circular spline fixed), the reduction ratio sign has to be changed.

5.2.1 Constant step

First, a constant voltage will be applied. In this case, the ESCON was configured with offset for both sides direction, so the 2.5 volts correspond to the motor stopped value.

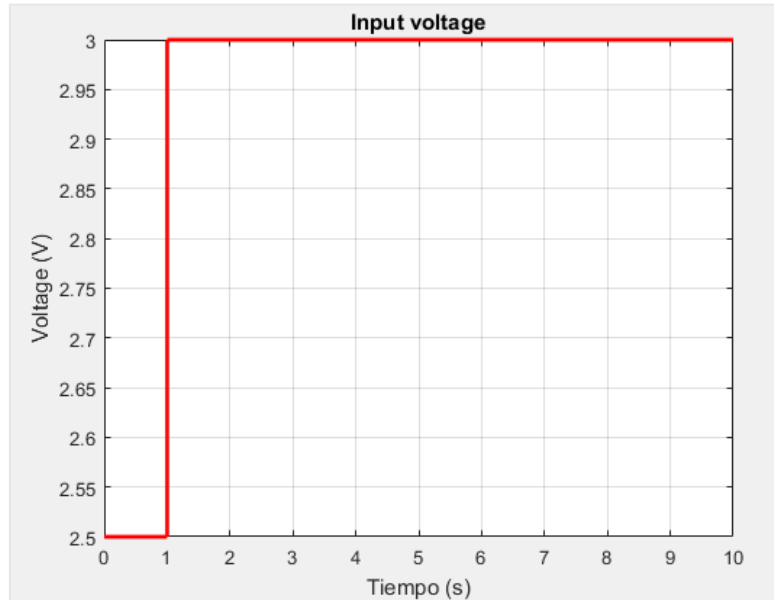


Figure 78. Constant input voltage.

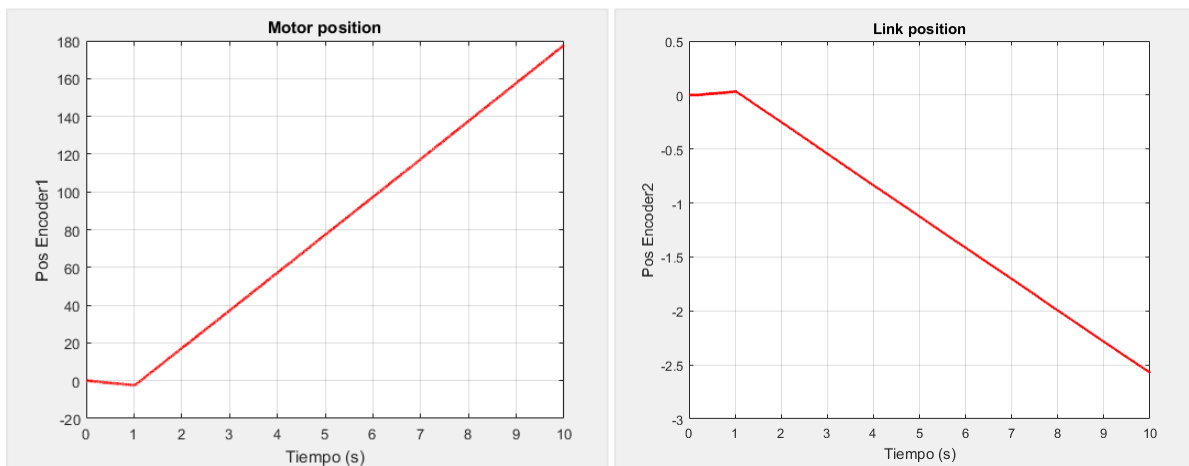


Figure 79. Motor and link positions for constant input voltage.

5.2.2 Stairs

For the following experiment, random stairs in positive and negative direction are given as input, having the same ESCON configuration for the motor stopped value.

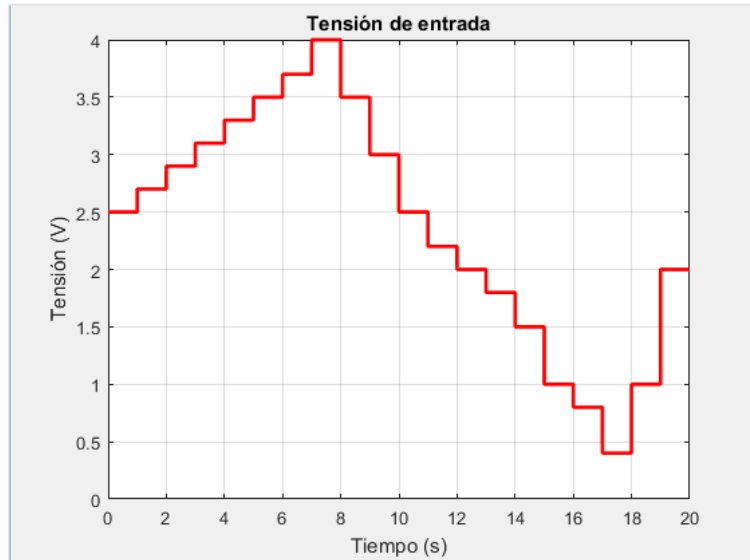


Figure 80. Stairs input voltage.

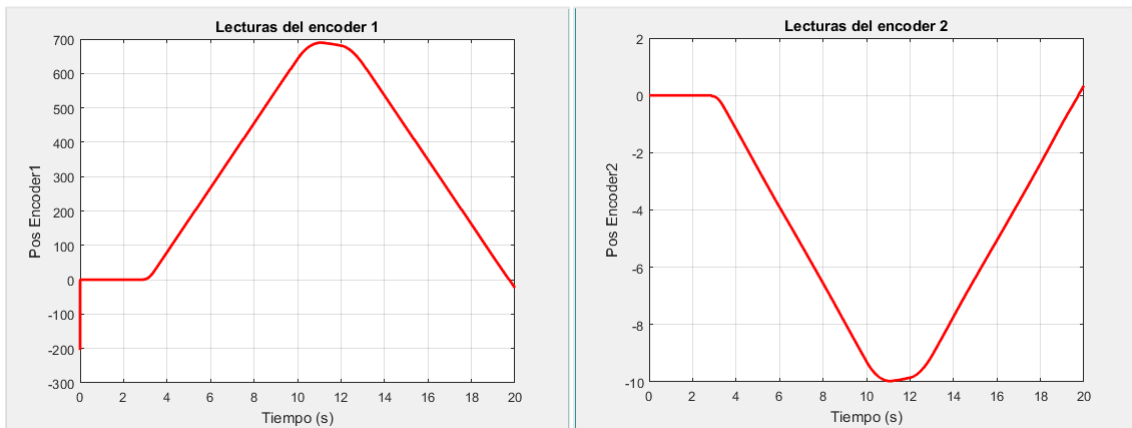


Figure 81. Motor and link position for stair input voltage.

5.2.3 Sinusoidal

Four sinus with frequencies 0.25, 0.5, 2, 5.

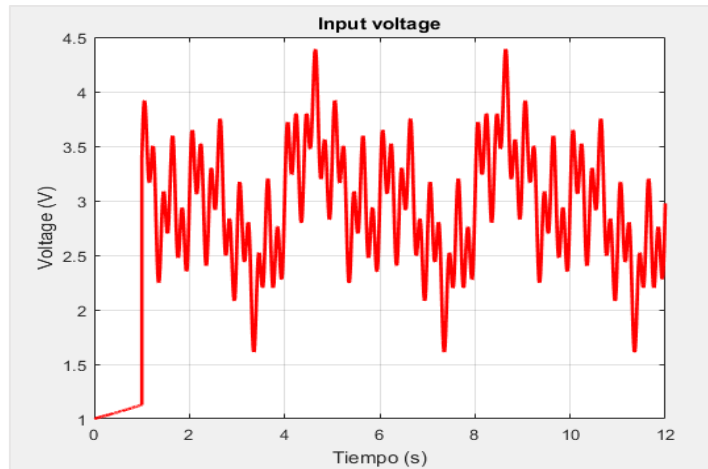


Figure 82. Sum of sinusoidals input voltage.

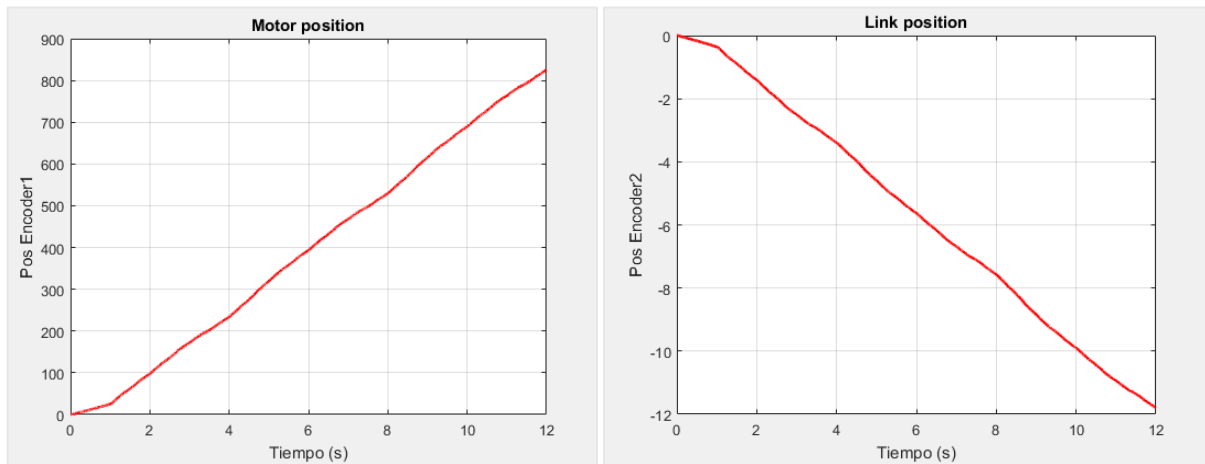


Figure 83. Encoder 1 and 2 measurements for the given experiment.

Table 13. Estimated reduction ratio from the three experiments.

Input signal	Estimated reduction ratio
Step	69.1322
Stairs	69.1577
Sinus	69.7731

5.3 Voltage control identification

5.3.1 Static friction parameters

Given the step input, the following results are obtained with $V_{in} = [1 \ 2 \ 3 \ 4 \ 5]$ since the input range is [0V-5V].

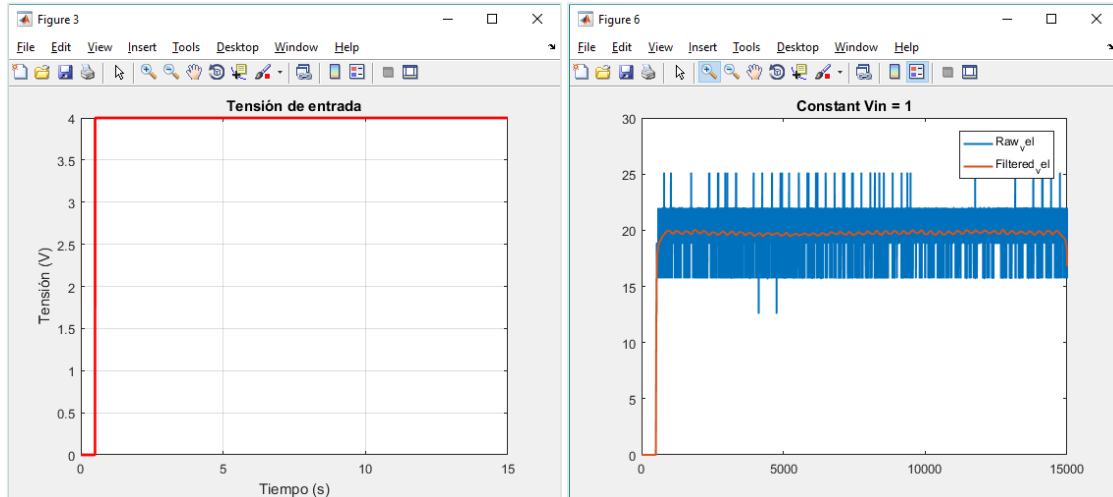


Figure 84. Input step and velocity response.

From the experimental data above a linear approximation can be made, if the different input values are plot against its respective output values the linear behaviour can be appreciated.

First way to determine the static friction range is to analyse the velocity in steady state, this mean given a constant input at different values inside the input range value [-2.5 , 2.5]V measure the steady state velocity and plot the following graph.

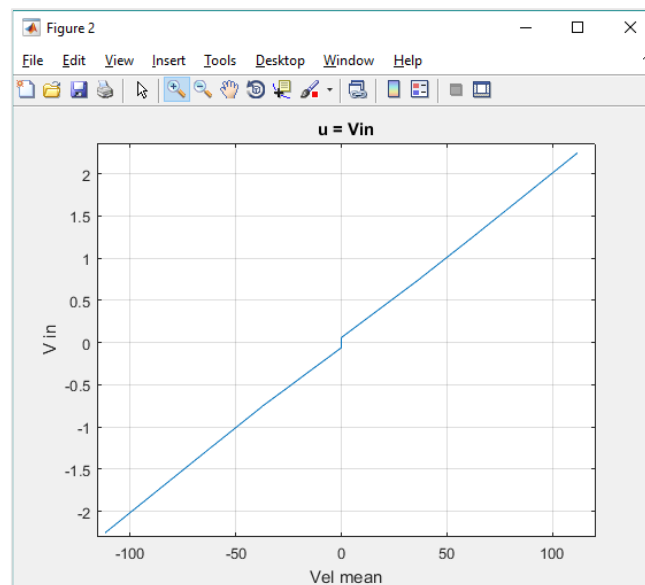


Figure 85. Steady state velocity vs. DAC Voltage input.

After obtaining the steady state velocity response the linear fit is applied at both sides (positive and negative) and then obtain the coefficients of the straight lines.

r_fit_positive = Linear model Poly1:

$$r_fit_positive(x) = p1*x + p2$$

Coefficients (with 95% confidence bounds):

$$p1 = 0.01978 (0.01918, 0.02038)$$

$$p2 = 0.02628 (-0.01275, 0.06532)$$

r_fit_neg = Linear model Poly1:

$$r_fit_neg(x) = p1*x + p2$$

Coefficients (with 95% confidence bounds):

$$p1 = 0.01984 (0.01921, 0.02046)$$

$$p2 = -0.02435 (-0.06497, 0.01626)$$

From the equations above, we can obtain the values of the friction parameters taking into account the model:

$$u_{DAC_{ss}} = U_{ss}K_{in} \quad (5.16)$$

$$U_{ss} = \left(R \frac{B_m}{K_m} + K_w\right) \dot{\theta}_{ss} + \left(\frac{R}{K_m}\right) \tau_c \quad (5.17)$$

$$p1 = \left(R \frac{B_m}{K_m} + K_w\right) / K_{in} \quad (5.18)$$

$$p2 = \frac{R}{K_m K_{in}} \quad (5.19)$$

As long as the parameters obtained from the linear fit above have some tolerance the calculations are made also for the higher and lower values obtained also, it is implemented in the code shown below:

Modelling, identification and control of an elastic joint.

```
% Both directions
%negative
p1_n = 0.01984;
p2_n = -0.02435;
p1_n_max = 0.02046;
p1_n_min = 0.01921;
p2_n_max = 0.01626;
p2_n_min = -0.06497;

%positive
p1_p = 0.01978;
p2_p = 0.02628;
p1_p_max = 0.02038;
p1_p_min = 0.01918;
p2_p_max = 0.06532;
p2_p_min = -0.01275;

%datasheet parameters
Km = 0.217; % (Datasheet) [Nm/A]
Kw = 1/4.6076; % Back emf (Datasheet) [rad/Vs]
R = 2.3; % Armature resistance (Datasheet) [Ohms]
Kin = 36/2.5; % From u(Disc DAC) to U(Voltage in the armature)
```


Modelling, identification and control of an elastic joint.

```

%% FROM NEGATIVE SIDE
bm_n = (p1_n*Kin-Kw)*Km/R;
tc_n = (p2_n*Kin*Km)/R;

bm_n_min = (p1_n_min*Kin-Kw)*Km/R;
tc_n_min = (p2_n_min*Kin*Km)/R;

bm_n_max = (p1_n_max*Kin-Kw)*Km/R;
tc_n_max = (p2_n_max*Kin*Km)/R;

% %% FROM POSITIVE SIDE
bm_p = (p1_p*Kin-Kw)*Km/R;
tc_p = (p2_p*Kin*Km)/R;

bm_p_min = (p1_p_min*Kin-Kw)*Km/R;
tc_p_min = (p2_p_min*Kin*Km)/R;

bm_p_max = (p1_p_max*Kin-Kw)*Km/R;
tc_p_max = (p2_p_max*Kin*Km)/R;

bm_pos = [bm_p, bm_p_min, bm_p_max]
bm_neg = [bm_n, bm_n_min, bm_n_max]
tc_pos = [tc_p, tc_p_min, tc_p_max]
tc_neg = [tc_n, tc_n_min, tc_n_max]

```

Which gives the following results:

Table 14. Friction parameters identified.

Parameter	Value	Min	Max
Bm positive	0.0064	0.0056	0.0072
Bm negative	0.0065	0.0056	0.0073
τ_c positive	0.0357	-0.0173	0.0887
τ_c negative	-0.0331	-0.0883	0.0221

Now, the viscous friction value obtained can be used as initialization value in the Idnlgrey identification procedure, despite this values have been identified already, its value might change in later identification procedures, where this parameters are included and interact with other model parameters.

5.3.2 Rigid body parameters

From the previous step, two parameters have been identified, the friction parameters identified using the steady state response of the system will be used to initialize the value of the parameters in the following step. Now the objective is to find the total inertia of the system, which is given by:

$$J_t = J_m + \frac{J_e}{N^2} \quad (5.17)$$

In other types of identification experiments discussed in the paper by Wernholt and Gunnarsson the from the overall inertia, the motor side and link side can be calculated if a scaling factor is defined as follows:

$$J_m = J_{jm}J_t ; J_e = J_{je}J_t \quad (5.18)$$

In the following steps, after identifying the total inertia, this scaling will be used to determine the value of the motor and link side inertia.

This is the conventional gear transmission inertia in the mechanical equation of the system; to identify the total inertia of the Harmonic drive system the input should not have high frequencies in order to avoid the elasticity excitation, so the reduced model torque equation is:

$$\ddot{\theta}(t) = \frac{(K_a i(t) - \tau_c) - B_m \dot{\theta}(t)}{J_t} \quad (5.19)$$

In the given context with the input being the voltage, the equation has to be rewritten, and keep all the units in terms of voltages instead torques, the equation rewritten is:

$$\ddot{\theta}(t) = \frac{\left(K_{in}u(t) - \frac{\tau_c R}{K_a}\right) - \left(K_w + \frac{B_m R}{K_a}\right)\dot{\theta}(t)}{\frac{J_t R}{K_a}} \quad (5.20)$$

Therefore, when estimating the model, the parameters calculated will be:

$$\mathbf{p}_1 = \frac{J_t R}{K_a}; \mathbf{p}_2 = K_w + \frac{B_m R}{K_a}; \mathbf{p}_3 = K_{in}; \mathbf{p}_4 = \frac{\tau_c R}{K_a}$$

In addition, knowing the values $R = 2.3$ and $K_a = K_w = 0.217$ from the motor datasheet, we can obtain the physical parameters:

$$J_t = \frac{p_1 K_a}{R} = [\mathbf{Kgm}^2]; \mathbf{B}_m = (p_2 - K_w) \frac{K_a}{R} = [\mathbf{Nms/rad}]; \tau_c = \frac{p_4 K_a}{R} = [\mathbf{Nm}]$$

As long as K_{in} is the relation between the DAC voltage and the voltage in the armature of the motor, this parameter is dimensionless.

Knowing the values of B_m and τ_c from the previous step, we can get an initial guess for parameters p_2 and p_4 , and knowing from the BLDC motor datasheet the motor inertia

$$J_{m_{DS}} = 0.000306 [\mathbf{Kgm}^2]$$

We can set the boundaries in the initial guess of the parameter values, this motor inertia from the datasheet does not correspond to the value of the motor side inertia in the model because the Harmonic drive wave generator inertia has to be added as well as the vibration reducer attached to

the motor axis. This value from the datasheet, scaled to voltage control by means of the motor constants can be set as a lower boundaries as long as it's the minimum value the total inertia will have. If during the identification procedure the total inertia identified is close or equal to this value the identification procedure should be discarded and calculated again with different data sets or different initial states, until a good fit to the data is approached with a "logical" value of this parameter.

Input signal

Now that the model has been defined for this step, the input waveform has to be chosen. As stated above the input must contain low frequencies in order to avoid flexibility excitation, this is necessary to determine the total inertia of the system. So for generating the 3 data sets a single sinus wave with different frequencies and the same amplitude is used, this sinusoidal has an offset to avoid the coulomb friction region, as long as we want to determine the dynamical properties of the system and the coulomb friction term has been identified before, as an example the velocity plot of the first data set is shown in the figure below, which has been obtained with an input equation as follows:

$$u_{DAC} = A\sin(2\pi ft) + Offset \quad (5.20)$$

Where the value of the frequency has been changed from one experiment to another.

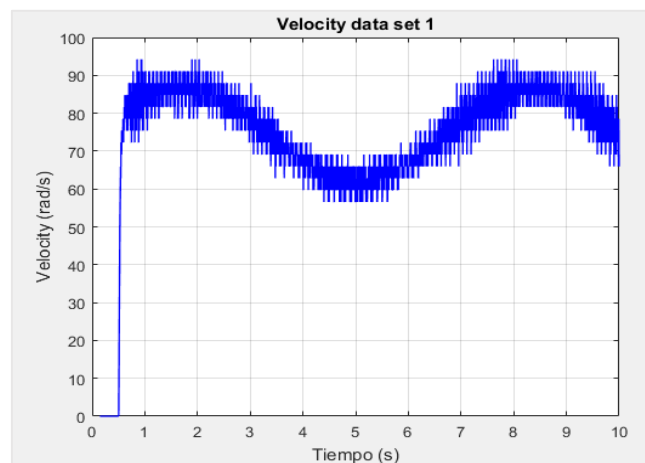


Figure 86. Velocity plot of the data set 1.

Model implementation

Once the data sets are collected, the next step would be to implement the reduced model in a MEX file, defining the parameters and the dynamic equation of the model containing these. This model will be called later by the identification script and will be put into an IDNLGREY object together with the initial states and parameters, the code of the MEX file used is in the annexes. When using high frequencies for the identification, the model will be changed to the Harmonic drive simplified model first, then more equations will be added in order to try to improve the response of the identified compared to the simulated response.

Identification procedure

First of all, the data sets have to be loaded, when acquiring the data with the Teraterm the matrix containing all the information is renamed, so the different data sets are saved in the matrix A, B and C and loaded into matlab with the following script. Remark that only the velocity is taken from the matrix even though all the measurements from the link side encoder also have been recorded only the motor side values of the velocity will be used.

```
%% This script needs functions: corregir_vel and saturacionencoder
addpath('C:\Users\Rafael C\Desktop\Ident_low_freq')
%% REMEMBER TO CHANGE THE NAME OF THE MATRIX CONTAINING THE DATA IN THE
FILES (A B C)...
% A
run ('lowfreq0075');
% B
run ('lowfreq_025');
% C
run ('LF_test1_2');
%Vel Filter design (filtfilt -> zero phase digital filter)
[b,a] = butter(5,0.3,'low');
vell_mdfiltval = 1;
%Encoder saturation function parameters
pos1off = 205.9;
pos2off = 205.8811;
threshold = 5000;
Ts = 0.001;
% EXTRACT DATA FROM TERATERM
%% Data set 1
Posencoder1 = A(:,1);
vell_noise = A(:,3);
tension_s = A(:,5);
tiempo = A(:,6);
% Encoder 1 motor position correction
v_aux1 = corregir_vel(Posencoder1,Ts);
encoders1 = saturacionencoder(Posencoder1,v_aux1,pos1off,threshold);
%Motor velocity filtering
vell_f = filtfilt(b,a,vell_noise);
vell = medfilt1(vell_f,vell_mdfiltval);
```

Only the script applied to the first data set is shown in the script, but the procedure is the same for data set 2 and 3, only changing the name of the matrix and the values where the signals are stored with the subscript b and c.

This script is included in the overall identification script, where all the information of the model is collected (this includes names of the inputs, states, parameters as well as its units and initial states), also an IDDATA object is built which contains all the input-output pairs for the three experiments, also when changing from the estimation of one data set to another, the previous values of the parameters are saved and used as an initial value for the following validation steps.

Results

At the end of the script, based on the estimated parameters, the following physical parameters are obtained:

Table 15. Parameters identified in the first try.

Jt	Bm	Kin	τ_c
7.7311e-04	0.0059	14	0.033

As a first conclusion, knowing the initial value of the gain input parameter was set randomly and it converged to a close value of which it is supposed to be $K_{in} = \frac{36}{2.5} = 14.4$ we can repeat the procedure, giving a tight tolerance around this value and estimating the parameters again.

Table 16. Parameters identified in the second try.

Jt	Bm	Kin	τ_c
7.6788e-04	0.0059	14.4	0.0330

Which gives a similar value for all the parameters, anyway when introducing the initial value for this parameter in the following steps, the voltage transformation should be conserved as long as the system is still working in voltage control, so we can state that the initial value for these parameters in the next step will be:

Table 17. Parameters used in the next step.

JtRK	BmRK	Kin	τ_cRK
0.00823878	0.289754	14.9999	0.37

After obtaining these values, the simulated response is compared to the real data and a % fit is given, as long as the velocity is estimated through numerical differentiation and knowing there are also some

oscillations due to the BLDC motor phases commutation the % fit won't be close to 100 but still will be good as it can be shown in the following figure:

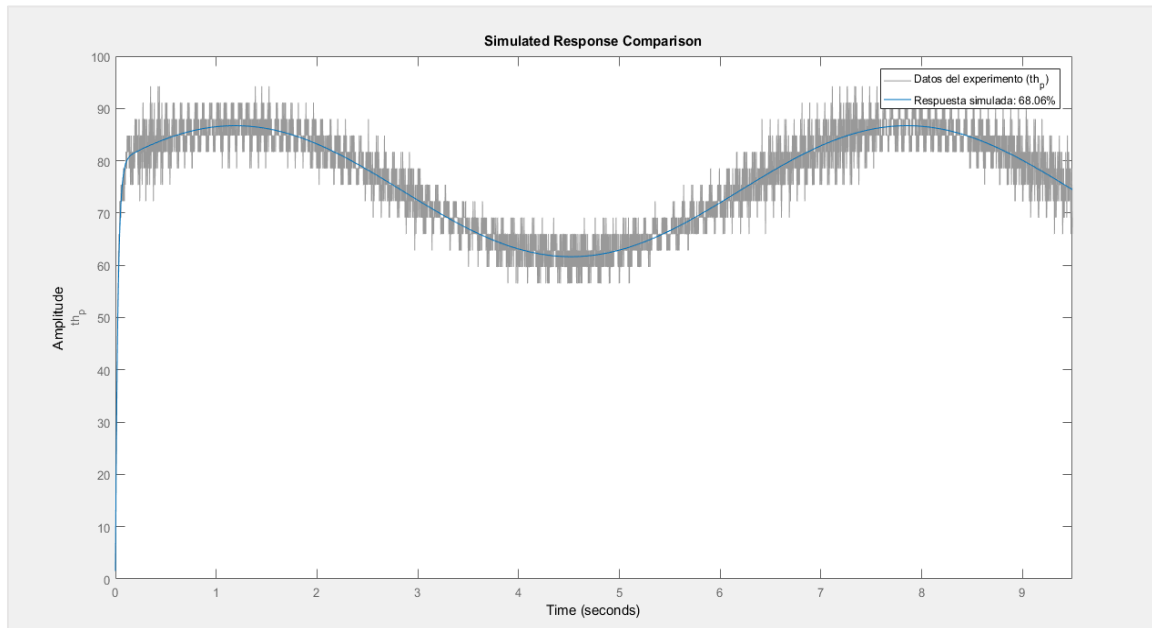


Figure 87. Simulated response vs. real data.

The fit is 68.06% but the simulated response follows very well the velocity data.

For dynamic systems, we can also examine the prediction errors using PE. We do this to see if the residuals seem to have a random nature:

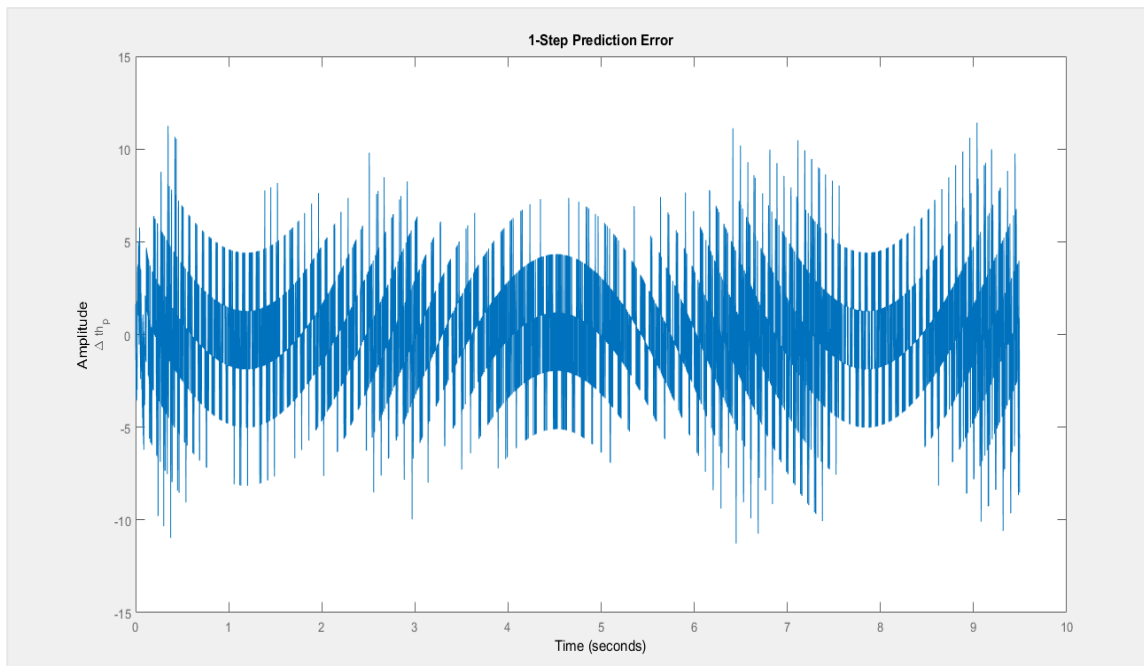


Figure 88. One-step prediction error.

Modelling, identification and control of an elastic joint.

As can be seen in the obtained plot, there is some kind of sinusoidal waves plus noise, the main reasons for these are:

- Signal obtained by numerical differentiation

This is the main reason why the prediction error has this kind of "bars" around the sinusoidal wave, as long as the encoder has a small quantization error when obtaining the velocity from the actual value and the value in the previous instant and the transformation from pulses to radians this "bars" have $n\pi$ amplitude as shown below:

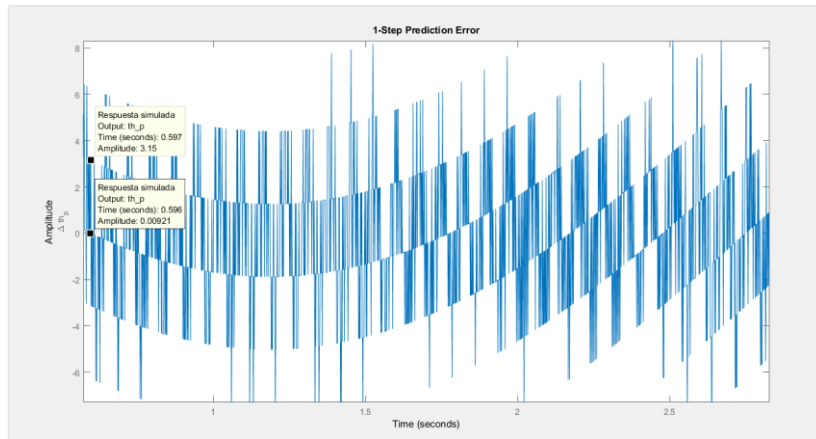


Figure 89. Amplitude of one of the "bars" equal to π .

- Simplified motor model

The BLDC motor has three phases, this means there is a commutation between phases as explained in previous points, the back EMF in this kind of motors can be trapezoidal or sinusoidal, if the simplified model of a DC motor has been used instead, this periodic shape of the back EMF force is left unexplained in the model.

- Noise error

There is always a small error in the DAC signal generator, the measurement equipment, etc...

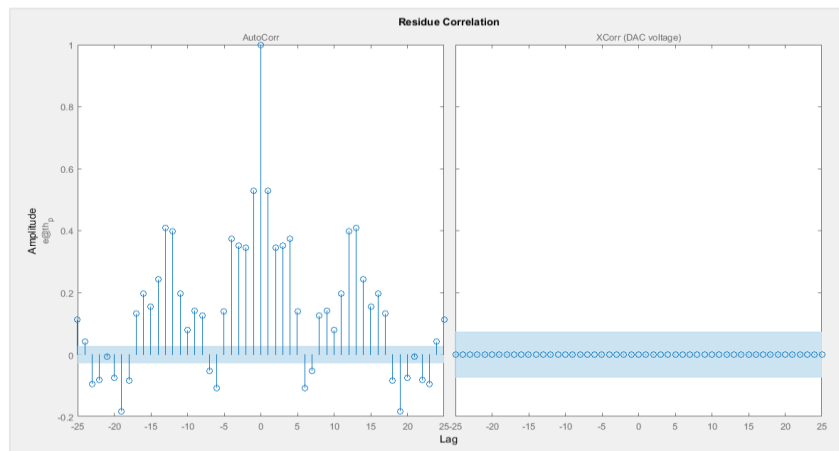


Figure 90. Residuals of the model estimated.

Summarizing, the NLGR object estimated has the following properties and values:

Continuous-time nonlinear grey-box model defined by 'Reduced_mod' (MEX-file):

$$\frac{dx}{dt} = F(t, u(t), x(t), p1, \dots, p4)$$

$$y(t) = H(t, u(t), x(t), p1, \dots, p4) + e(t)$$

with 1 input, 1 state, 1 output, and 4 free parameters (out of 4).

Input:

u(1) DAC voltage(t) [V]

State: initial value

x(1) th_p(t) [rad/s] xinit@exp1 0 (fix) in [-Inf, Inf]

Output:

y(1) th_p(t) [rad/s]

Parameters: value standard dev

p1 Jt*R/Ka 0.00823878 0.0377831 (est) in [0.00803878, 0.00823878]

p2 Velocity term (Kw + R/Ka*Bm) 0.289754 1.33057 (est) in [0.22, 0.38]

p3 Kin 14.4 66.1204 (est) in [14.4, 18]

p4 tcRK 0.37 1.68788 (est) in [0.0028, 0.37]

Status:

Termination condition: Maximum number of iterations reached.

Number of iterations: 10, Number of function evaluations: 75

Estimated using Solver: ode45; Search: lm on time domain data "Datos del experimento".

Fit to estimation data: 80.59%

FPE: 8.258, MSE: 8.247

Now we have a good initial value estimate for these parameters that will be used in the next steps in order to

5.3.3 Harmonic drive (simplified model) parameter identification

The following step requires that the data should be informative enough for open loop operation. This means that the input should be persistently exciting of at least $\frac{n_p}{2}$ different frequencies, where n_p is the number of parameters to be identified in the system. This leaves a good amount of freedom for the input choice, as L.Ljung stated in [27], the input must have limited amplitude and be periodic, this may have certain advantages when obtaining the data set to estimate the parameters in an open loop system. In addition, the sampling rates have some considerations to be made, as follows:

- Very fast sampling lead to numerical problems, model fits in high-frequency bands, and poor returns for extra work.
- As the sampling time increases over the natural time constants of the system, the variance increases drastically.
- Optimal choices of T for a fixed number of samples will lie in the range of the time constants of the system. These are, however not exactly known, and overestimating them may lead to very bad results.

From the considerations above it can be concluded that a sampling frequency that is about ten times the bandwidth of the system should be a good choice in most cases. Note that this discussion concerns the sampling rate chosen for the model building. With “cheap” data acquisition, we can always sample as fast as possible during the experiment and leave the actual choice of T for later by digitally pre-filtering and decimating the original data record.

Input signals

The inputs consist in the sum of 3 sinus and 1 cosines, the frequencies used are described in the table below. The offset and the amplitude is adjusted in order to keep the motor working in the positive side or both directions, knowing that the motor stop corresponds to 2.5V and boundaries at [0V- 5V], the part of the code describing the equation of the input is the following:

```
float u = 1.75*(0.5*sin(2*3.1415*tiempo*5) + sin(2*3.1415*tiempo*0.75)+  
sin(2*3.1415*tiempo*0.5) + 0.75*cos(2*3.1415*2*tiempo) )/5 + 3.75;
```

To make easier the description of the identification procedure, the different data sets used in order to identify the parameters are described in a table and some plots are shown where the frequencies as well as the amplitudes of the resulting signals can be appreciated.

Table 18. Data set properties.

Data set “sins1”	
Frequencies f_1, f_2, f_3, f_4 (Hz)	0.5, 0.75, 1, 5
Offset (V)	3.75
Amplitude (V)	1.13

```
float u = 1.75*(0.5*sin(2*3.1415*tiempo*0.75) + sin(2*3.1415*tiempo*4)+
sin(2*3.1415*tiempo*5) + 0.75*cos(2*3.1415*2*tiempo) )/5 + 3.75;
```

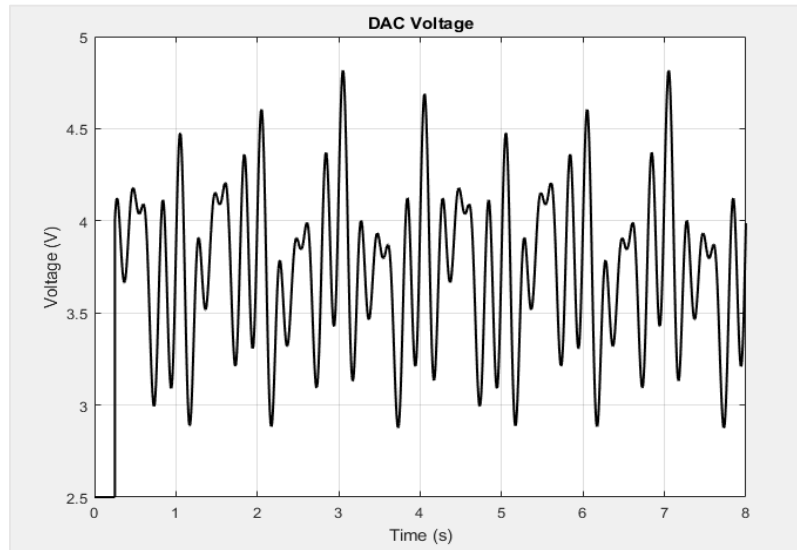


Figure 91. Input signal plot for data set "sins1".

Table 19. Data set properties.

Data set "scs1"	
Frequencies f_1, f_2, f_3, f_4 (Hz)	0.5, 0.75, 2, 5
Offset (V)	3.75
Total amplitude (V)	1.13

```
float u = 1.75*(0.5*sin(2*3.1415*tiempo*5) + sin(2*3.1415*tiempo*0.75)+
sin(2*3.1415*tiempo*0.5) + 0.75*cos(2*3.1415*2*tiempo) )/5 + 3.75;
```

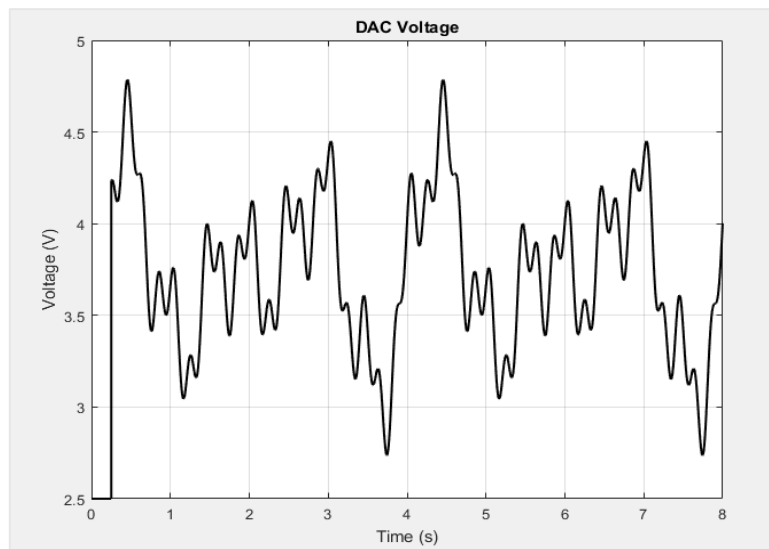


Figure 92. Input signal plot for data set "scs1".

Table 20. Data set properties.

Data set "scs2"	
Frequencies f_1, f_2, f_3, f_4 (Hz)	0.5, 2, 5, 15
Offset (V)	3.75
Total amplitude (V)	0.77

```
float u = (sin(2*3.1415*tiempo*5) + 5*sin(2*3.1415*tiempo*15)+ sin(2*3.1415*tiempo*0.5) + 0.75*cos(2*3.1415*2*tiempo) )/10 + 3.75;
```

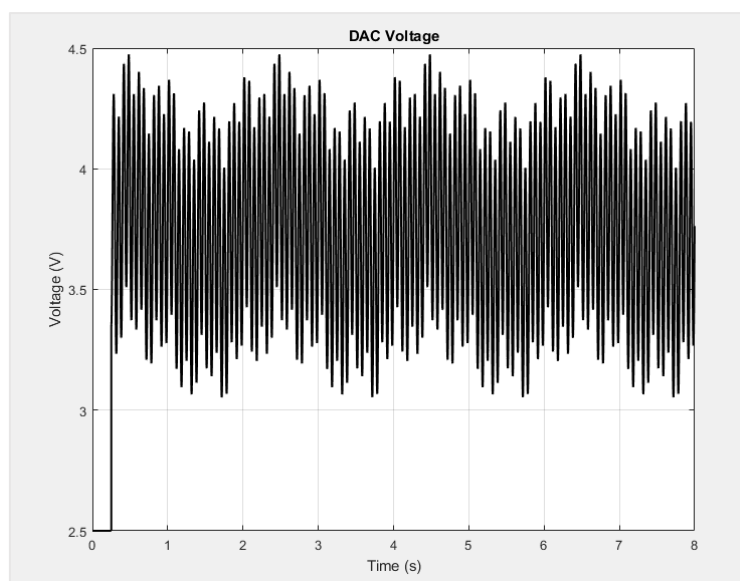


Figure 93. Input signal plot for data set "scs2".

Table 21. Data set properties.

Data set "bdsins"	
Frequencies f_1, f_2, f_3, f_4 (Hz)	0.75, 2, 4, 5
Offset (V)	2.5
Total amplitude (V)	1.62

```
float u = 2.5*(0.5*sin(2*3.1415*tiempo*0.75) + sin(2*3.1415*tiempo*4)+
sin(2*3.1415*tiempo*5) + 0.75*cos(2*3.1415*2*tiempo) )/5 + 2.5;
```

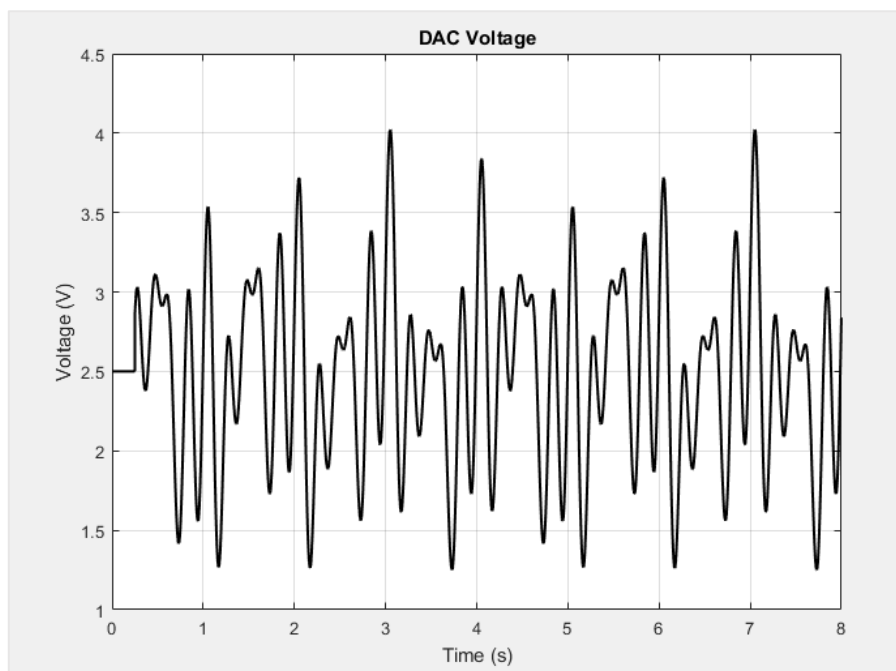


Figure 94. Input signal plot for data set "bdsins".

For now, on when describing the identification procedure, the data sets will be named as presented in the present point, the last data set will be used to identify the hysteresis, that is appreciated in the kinematic error signal.

All the data sets are recorded with the ESCON power stage working in voltage control, this means the models used are the ones described in section 3 once the parameters are identified in voltage control the values have to be transformed in order to keep the units of the S.I, the most usual way to model robots are using the torques equations which implies that the measurements are the input torque, or, for an electrical motor the current multiplied by its torque constant, in this case the "torque" parameters are calculated after the values of the parameters are obtained.

Model implementation

Now the model used for the parameter estimation will be the simplified Harmonic drive model, with 12 parameters, where five of them are known from datasheet information or obtained in previous steps, one input, four states and four outputs and it is given by the following equations:

$$\ddot{\theta}(t) = \frac{\left((u_{DAC}(t)K_{in} - \frac{\tau_c R}{K_a}) + \frac{\left(\frac{B}{N} \left(\dot{q}(t) - \frac{\dot{\theta}(t)}{N} \right) + \frac{K}{N} \left(q(t) - \frac{\theta(t)}{N} \right) \right) R}{K_a} - (K_w + \frac{B_m R}{K_a}) \dot{\theta}(t) \right)}{J_m K_a / R} \quad (5.22)$$

$$\ddot{q}(t) = - \frac{\left(b_e \dot{q}(t) + K \left(q(t) - \frac{\theta(t)}{N} \right) + B \left(\dot{q}(t) - \frac{\dot{\theta}(t)}{N} \right) \right)}{J_e} \quad (5.23)$$

This model has been implemented in a MEX file for a grey box model identification.

Some things have to be clarified for this model:

Voltage control: Motor equation has to be changed from torque variables to voltage variables, only for the motor side equation, the link side equation as long as the input is the elastic torque, no units transformation are needed.

$$V = IR + \frac{Ldi}{dt} + E \quad (5.24)$$

To demonstrate the effect voltage and torque have on speed, we are only concerned with steady-state behaviour at the moment, which implies that the current is constant, so the inductor L can be removed and there is no change in speed, which means the torque produced by the motor and the torque produced by the load must be equal.

The second assumption implies that the torque is produced by the motor as given in the following equation:

$$\tau_m = K_t \varphi i \quad (5.25)$$

Where K_t is a constant inherited from its internal design, φ is the total flux and τ_m is the load torque in steady state, rearranging for i and including it in the previous equation:

$$V = \frac{\tau_m}{K_t \varphi} R + E \quad (5.26)$$

Now the EMF, which is dependent upon the total flux motor specific factors that make a second constant K_w and the speed of the motor n_{mot} .

$$E = K_w \varphi n \quad (5.27)$$

Which gives:

$$V = \frac{\tau_m}{K_t \varphi} R + K_w \varphi n \quad (5.28)$$

The equation above actually represents a linear motor; in adapting this to an angular rotating motor, we consider the flux constant at its full value. In doing so it is combined with each constant to produce the torque constant and electrical constant of the motor, denoted K_a, K_w in the model.

$$V = \frac{\tau_m}{K_a} R + K_w \dot{\theta} \quad (5.29)$$

Alternatively, equivalent:

$$\dot{\theta} = \frac{V}{K} - \frac{T}{K^2} R \quad (5.30)$$

This means that for a fixed load, the speed of the motor is affected by applied voltage so an increase in the voltage means an increase in the speed, on the other hand for a fixed voltage the speed of the motor is inversely affected by the load, so an increase in the load torque means a decrease in the speed.

- Linear elasticity: Only the linear spring stiffness coefficient will be identified here, assuming the elastic torque is linear with respect the torsional angle.
- No hysteresis effects modelled.
- DC motor simplified model instead EC (BLDC) 3 phase with periodic back EMF waveform, this might add noise and probably a pattern when analysing the 1 step prediction error of the simulated response.
- Simplified friction model, only coulomb term plus viscous friction.

The MEX file implemented containing the model as well as the IDNLGREY script are included in the annexes.

Identification procedure

This main script calls another script used for loading the data, where the position of the encoder signal is unwrapped, the velocity is estimated from the fixed position and then a phase delay filter is applied with the command "filtfilt", this gives different context for the data generated, given the following scenarios:

- Velocity estimated with noise
- Velocity filtered with no phase filter

The link side velocity is moving 70 times slower than the motor input, this means that for the given encoder resolution there is a quantization error in the measurements. So, when estimating velocity by

Modelling, identification and control of an elastic joint.

differentiation with position increments with fixed time ($T_s = 0.001$) the signal obtained are just pulses multiple of π , the script used as well as the velocities from the link side are shown.

```
%% This script needs functions: corregir_vel and saturacionencoder
addpath('C:\Users\Rafael C\Desktop\Ident_high_freq\newdata')
%% REMEMBER TO CHANGE THE NAME OF THE MATRIX CONTAINING THE DATA IN THE
FILES (A B C)...
run ('frequs3'); %% Check Ts = 0.001
run ('frequs2'); %% Check Ts = 0.001
%Vel Filter design (filtfilt -> zero phase digital filter)
[b,a] = butter(5,0.3,'low');
vell1_mdfiltval = 1;
vel2_mdfiltval = 30;
%Encoder saturation function parameters
pos1off = 205.9;
pos2off = 205.8811;
threshold = 5000;
Ts = 0.001;
% EXTRACT DATA FROM TERATERM
Posencoder1 = A(:,1);
Posencoder2 = A(:,2);
vell1_noise = A(:,3);
vel2_noise = A(:,4);
tension_s = A(:,5);
tiempo = A(:,6);
% Encoder 1 motor position correction
v_aux1 = corregir_vel(Posencoder1,Ts);
encoders1 = saturacionencoder(Posencoder1,v_aux1,pos1off,threshold);
% Encoder 2 link position correction
v_aux2 = corregir_vel(Posencoder2,Ts);
encoders2 = saturacionencoder(Posencoder2,v_aux2,pos2off,threshold);
%Motor velocity filtering
vell1_f = filtfilt(b,a,vell1_noise);
vell1 = medfilt1(vell1_f,vell1_mdfiltval);
%Link velocity filtering
vel2_f = filtfilt(b,a,vel2_noise);
vel2 = medfilt1(vel2_f,vel2_mdfiltval);
```

The function “corregir_vel” estimates the velocity with “jumps” produced by the wrapping of the

encoder measurements. The signal “v_aux” is used to detect this jumps and fix them with the function “saturacionencoder”. The velocity estimated online has the wrapping problem fixed already and can be filtered directly with the “filtfilt” command to apply a zero phase filtering, the medfilt can be used to see the shape of the sinus in the velocity link but it’s not used for identification. Now the identification script will be applied for the noisy data and the filtered one.

Identification 1, noise vs. filtfilt velocity 2 filter

In this section, the effect of the filter in the velocity 2 estimated signal in the identification procedure would be analysed, to do so the same data sets will be used to identify the parameters, first with noise velocity measurements, then with no phase-filtered velocity 2.

Table 22. Identification 1 procedure info.

Identification procedure 1 description	
Number of samples	7990
Input signal 1 (Estimation 1)	Scs2
Input signal 2 (Estimation 2)	Scs1
Input signal 3 (Validation)	Sins1
Velocity 2 filter	Filtfilt, noise
Model identified	HD simple
Number of outputs	4
Search methods (Estimation1, Estimation2)	lsqnonlin, lm

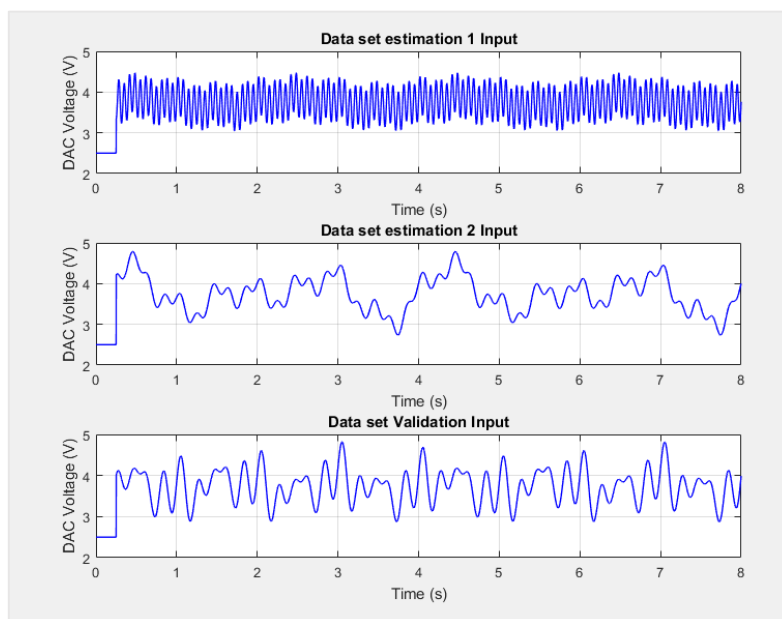


Figure 95. Input signals of the data sets.

Modelling, identification and control of an elastic joint.

With the given initial values.

Table 23. Initial parameters values and boundaries.

Parameter\Set values	x0	x0 lower	x0 upper
JmRK	0.00736543	0.0073	0.009
Je	0.05	0.01	1
N	70	-	-
BmRK	0.29927	0.19927	0.39927
D	1	0	50
Be	0.001	eps(0)	1
Kin	14.4	14.325	14.475
K1	2000	10	80000
τ_c RK	0.3007	0.28	0.315

The following parameters are obtained for the velocity signal with no filter set and the different estimation methods applied to the same data sets.

Data has 4 outputs, 1 inputs and 8991 samples.

Termination condition: Change in cost was less than the specified tolerance.

Number of iterations: 10, Number of function evaluations: 11.

Estimated using Solver: ode45; Search: lsqnonlin.

Table 24. Final estimated parameters for both velocity 2 filter data sets.

Search method = 'Lsqnonlin' , 'lm'		
Parameter\Value	xf, filtfit	xf , noise
JmRK	0.0073	0.0073
Je	0.0214815	0.021495
N	70	70
BmRK	0.284971	0.284975
D	0.0562333	0.0564026
Be	0.153757	0.153803
Kin	14.461	14.475
K1	40.4364	40.4293
τ_c RK	0.28	0.280456

With FPE: 1.028e-05, MSE: 3.786 for filtfit.

With FPE: 0.0002831, MSE: 5.654 for noise.

Modelling, identification and control of an elastic joint.

In the velocity 2 noise identification, the validation data, which is being compared, has been filtered with the “filtfilt” matlab command as shown below.

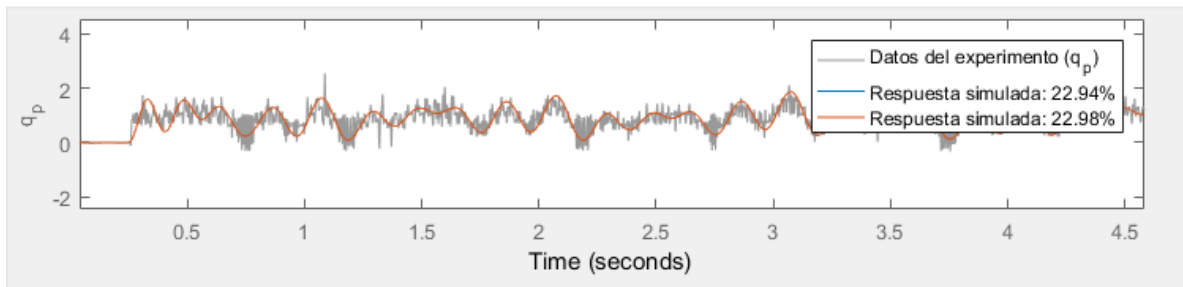


Figure 96. Simulated response vs. real data measurements.

After transforming the units of the parameters, the following physical parameters are obtained, the first data set is estimated with the search method “lsqnonlin”, with the higher input frequencies data set because this method updates the value of the elasticity parameters. Then the second data set for estimation and fit of the rest of the parameters for this purpose the “lm” search method has been chosen because gives the smallest cost after finishing all the iterations in the identification procedure.

Table 25. Final parameter value after transformation.

Parameter\Value	xf filtfilt	Xf noise
Jm	6.8874e-4	6.8874e-04
Je	0.0215	0.0215
N	70	70
Bm	0.0064	0.0064
D	0.0562	0.0564
Be	0.1538	0.1538
Kin	14.475	14.475
K1	40.4364	40.4293
τ_c	0.0265	0.0265
Jt	6.9312e-04	6.9313e-04

Modelling, identification and control of an elastic joint.

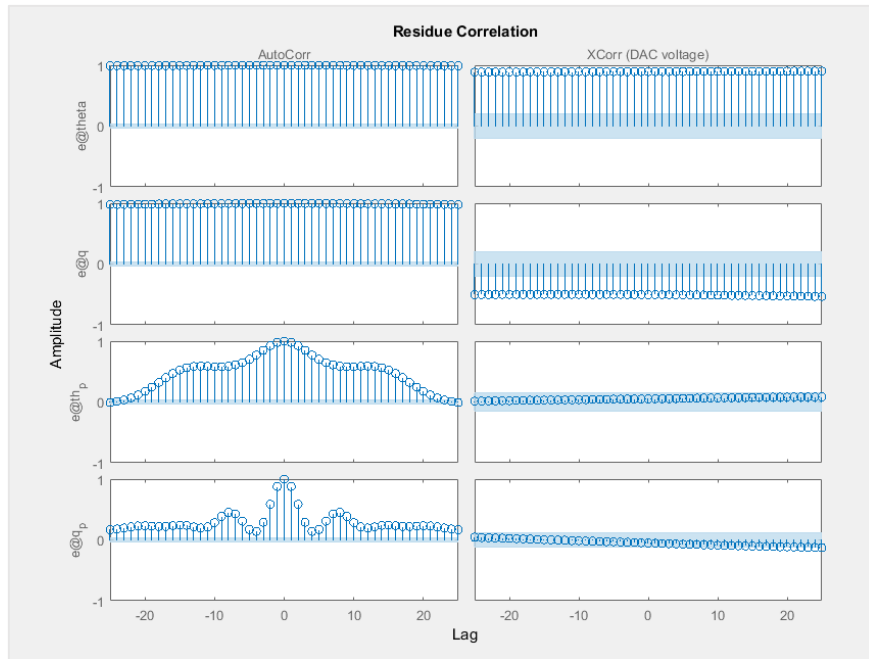


Figure 97. Residuals of the obtained model.

Identification 2, noise vs. filtfilt velocity 2 filter

Now the same identification procedure will be made but with different data sets orders, this will be made in order to check the consistency of the parameters estimated in the first identification.

Table 26. Identification 2 procedure info.

Identification procedure 1 description	
Number of samples	7990
Input signal 1 (Estimation 1)	Scs1
Input signal 2 (Estimation 2)	Scs2
Input signal 3 (Validation)	Sins1
Velocity 2 filter	Filtfilt, noise
Model identified	HD simple
Number of outputs	4
Search methods (Estimation1, Estimation2)	lsqnonlin, lm

Modelling, identification and control of an elastic joint.

With the same initial values used before.

Table 27. Identification 2 initial parameters and its boundaries.

Parameter\Set values	x0	x0 lower	x0 upper
JmRK	0.00736543	0.0073	0.009
Je	0.05	0.01	1
N	70	-	-
BmRK	0.29927	0.19927	0.39927
D	1	0	50
Be	0.001	eps(0)	1
Kin	14.4	14.325	14.475
K1	2000	10	80000
τ_c RK	0.3007	0.28	0.315

The following parameters are obtained for the different velocity 2 filter.

Table 28. Final estimated parameters for both velocity 2 filter data sets.

Search method = 'Lsqnonlin' , 'lm'		
Parameter\Value	xf, filtfilt	xf , noise
JmRK	0.00755584	0.00755577
Je	0.0259733	0.0260556
N	70	70
BmRK	0.286499	0.286499
D	15.0979	15.0953
Be	0.00025576	0.000251888
Kin	14.4181	14.4181
K1	6569.23	6568.81
τ_c RK	0.315	0.315

With FPE: 9.745e-07, MSE: 3.05 for filtfilt

With FPE: 2.737e-05, MSE: 4.818 for noise

When comparing the response of the estimated model with velocity 2 filter and without filterwin we got the same result as shown in figure 99 and 100.

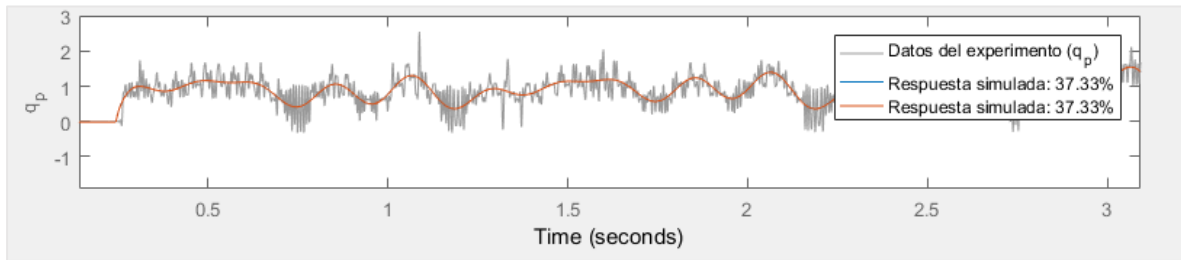


Figure 98. Noise velocity2 estimated response.

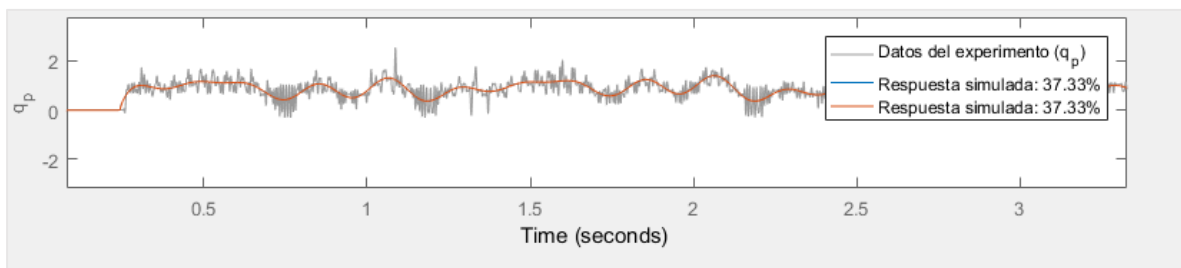


Figure 99. Filtfilt velocity2 estimated response.

After transforming the parameters, the following values are obtained.

Table 29. Final values obtained after transformation.

Parameter\Value	xf filtfilt	Xf noise
Jm	7.1288e-04	7.1287e-04
Je	0.0260	0.0261
N	70	70
Bm	0.0066	0.0066
D	15.0979	15.0953
Be	2.5576e-04	2.5189e-04
Kin	14.4181	14.4181
K1	6.5692e+03	6.5688e+03
τ_c	0.0297	0.0297
Jt	7.1818e-04	7.1819e-04

Modelling, identification and control of an elastic joint.

From the data obtained now, we can compare between the identification 1 and identification 2 results in the following table and make some conclusions.

Table 30. Results comparison between identification 1 and 2.

Parameter\Value	xf Ident 1	xf Ident 2
Jm	6.8874e-4	7.1287e-04
Je	0.0215	0.0261
N	70	70
Bm	0.0064	0.0066
D	0.0562	15.0953
Be	0.1538	2.5189e-04
Kin	14.475	14.4181
K1	40.4364	6.5688e+03
τ_c	0.0265	0.0297
Jt	6.9312e-04	7.1819e-04

From the data obtained the main conclusions are:

- Velocity 2 filter “filtfilt” does not change the value of the estimated model parameters.
- The search method applied gives different results depending on the frequency of the input signal.
- The values for the previous steps remain unchanged, so those values will be fixed for the following identification steps.
- The values identified that are similar independently of the order of the estimation data sets will remain unchanged too.

Now the main problem is to determine:

- Which input signal is more appropriate for identifying the elasticity parameters

How number of iterations affect to the final estimated values

With the same initial parameters value the estimation 2 identification will be carried out with an “lm” search method, first with seven iterations, then with 35. The initial parameter value used is let free without restrictions in all the values excepting Bm and tau_c.

Reducing the second searchmethod iterations to 7:

Parameters:	value	standard dev	
p1 Motor inertia [kgm ²]	0.00388188	0.26255	(est) in [3.36543e-05, 0.007]
p2 Link side inertia [kgm ²]	0.641145	43.3925	(est) in]0, 1]
p3 HarmonicDrive reduction ratio	70	0	(fix) in [70, 70]
p4 Motor viscous friction [Nms/rad]	0.319736	21.6272	(est) in [0.249278, 0.349278]
p5 Spring Damping [Nms/rad]	4.04852e-07	5.23149	(est) in [0, 50]
p6 Link viscous friction [Nms/rad]	4.94066e-324	0.720346	(est) in]0, 10]
p7 Current to torque motor constant [Nm/A]	16.1707	1093.85	(est) in [14.325, 19.475]
p8 Spring stiffness constant [Nm/rad]	13160.8	890858	(est) in [10, 80000]
p9 Coulomb friction	0.2875	19.45	(est) in [0.2875, 0.315]

With the second searchmethod iterations 35:

Parameters:	value	standard dev	
p1 Motor inertia [kgm ²]	0.00384738	0.266723	(est) in [3.36543e-05, 0.007]
p2 Link side inertia [kgm ²]	0.654684	45.4157	(est) in]0, 1]
p3 HarmonicDrive reduction ratio	70	0	(fix) in [70, 70]
p4 Motor viscous friction [Nms/rad]	0.319741	22.1684	(est) in [0.249278, 0.349278]
p5 Spring Damping [Nms/rad]	0	1.01122	(est) in [0, 50]
p6 Link viscous friction [Nms/rad]	4.94066e-324	0.962467	(est) in]0, 10]
p7 Current to torque motor constant [Nm/A]	16.171	1121.21	(est) in [14.325, 19.475]
p8 Spring stiffness constant [Nm/rad]	13160.8	913091	(est) in [10, 80000]
p9 Coulomb friction	0.2875	19.9364	(est) in [0.2875, 0.315]

From the results we can conclude that more iterations don't provide a better result in the second estimation data set, however when comparing the correlation residuals with the one obtained previously we can see that there is a small improvement in the cross correlation between the positions and the input as well as the autocorrelation of the velocity 2.

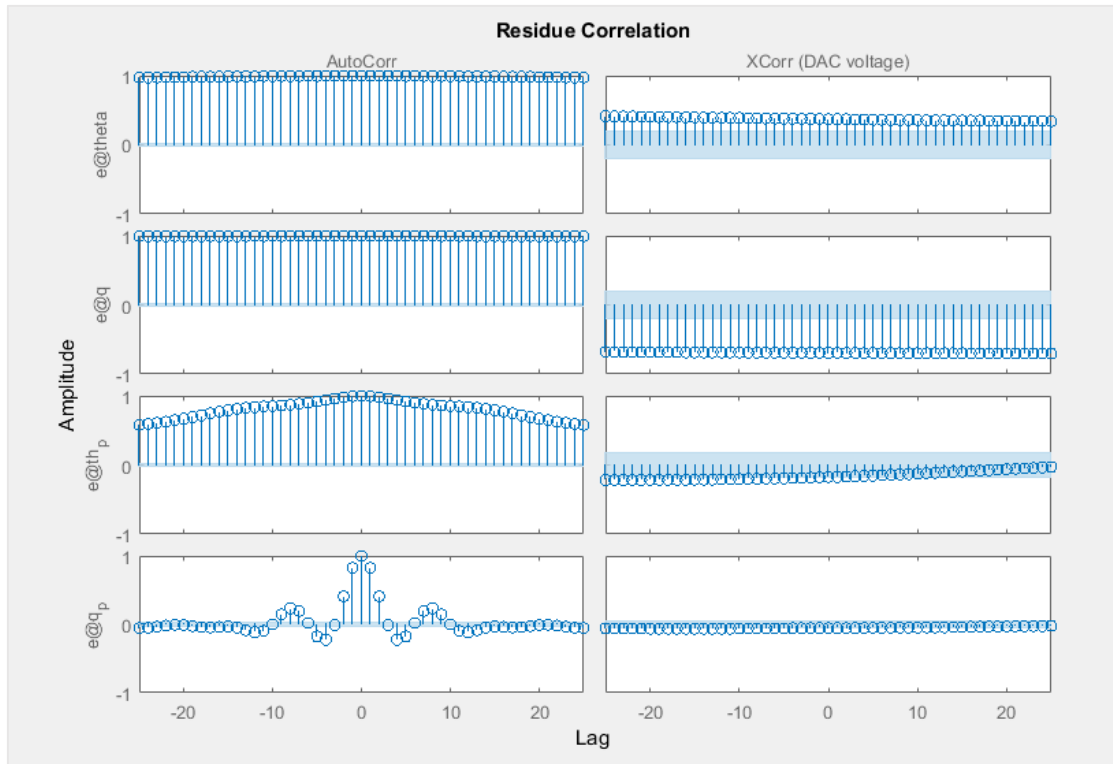


Figure 100. Residual correlation analysis for the identified models.

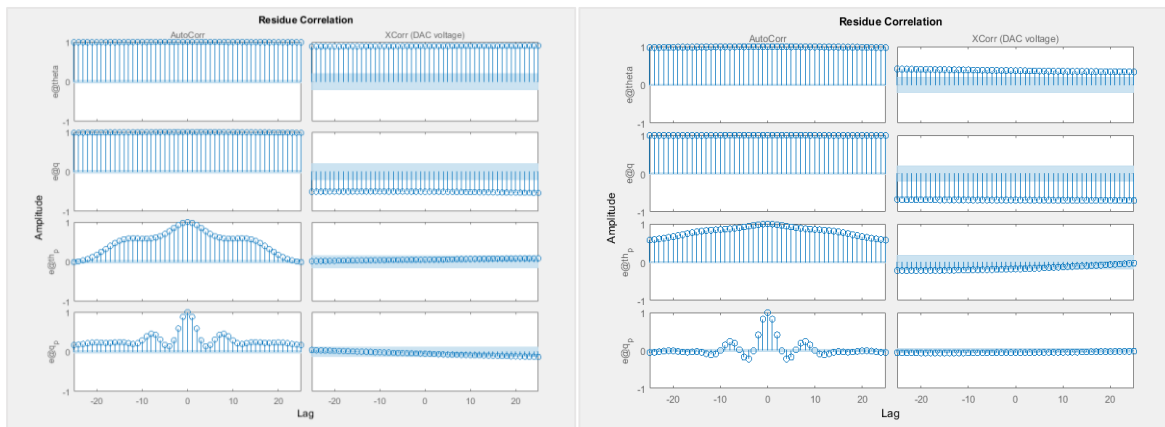


Figure 101. Left side residuals from first ident, Right side residuals from last ident.

When analyzing the residue correlation we detect a bad cross correlation in the previous identified model, even though the fit to data was good enough, now in the next step more phenomenas will be added to the model and the residuals of the model analyzed will be compared.

5.3.4 Harmonic drive with nonlinear friction and elasticity parameter identification

Until now all the identification procedures have been done with data running only in the positive direction, this mean that the voltage sent by the DAC was set between [2.5V – 5V]. So with the 2.5V in the DAC corresponding to the motor stopped, now some experiments will be carried out in both directions this implies zero velocity crossing. The simplified model used before with the Coulomb friction having only one sign results in an incorrect simulated response due to the fact that the model was considered only for the motor running in one direction. The response simulated when crossing zero velocities with the model identified in the previous section is shown in the figure below.

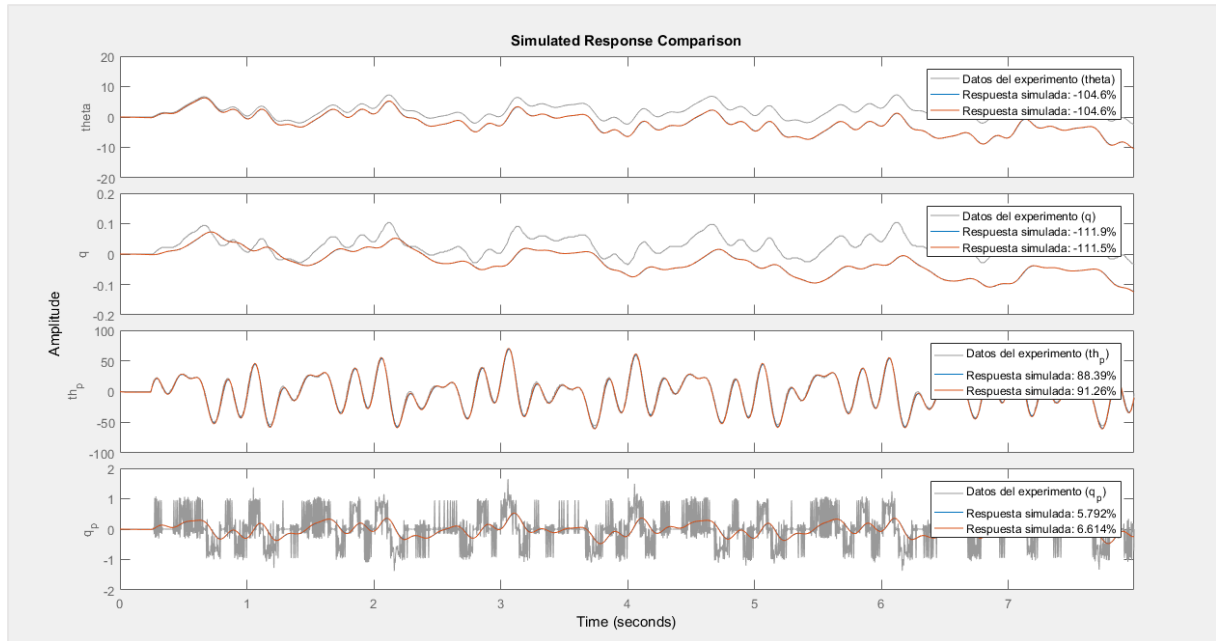


Figure 102. Identified NLGR with one direction simulated response compared with both directions data set.

Some considerations made above will change, being now:

- Nonlinear elasticity: The cubic polynomial for the elasticity will be introduced, as stated in equation (3.28).
- Nonlinear friction will be modeled through a trigonometric function, this is because other models add discontinuities at zero crossing, which makes numerical implementation a hard task, implemented in the following equation:

$$\tau_f = B_m \dot{\theta} + \left(\tau_c + \frac{F_s}{\cosh(\alpha \dot{\theta})} \right) \tanh(\beta \dot{\theta}) \quad (5.31)$$

So for this model the inputs used will cross zero velocity in order to identify the friction phenomena that arises when the system runs at zero or very small velocities.

For this purpose the following inputs are used:

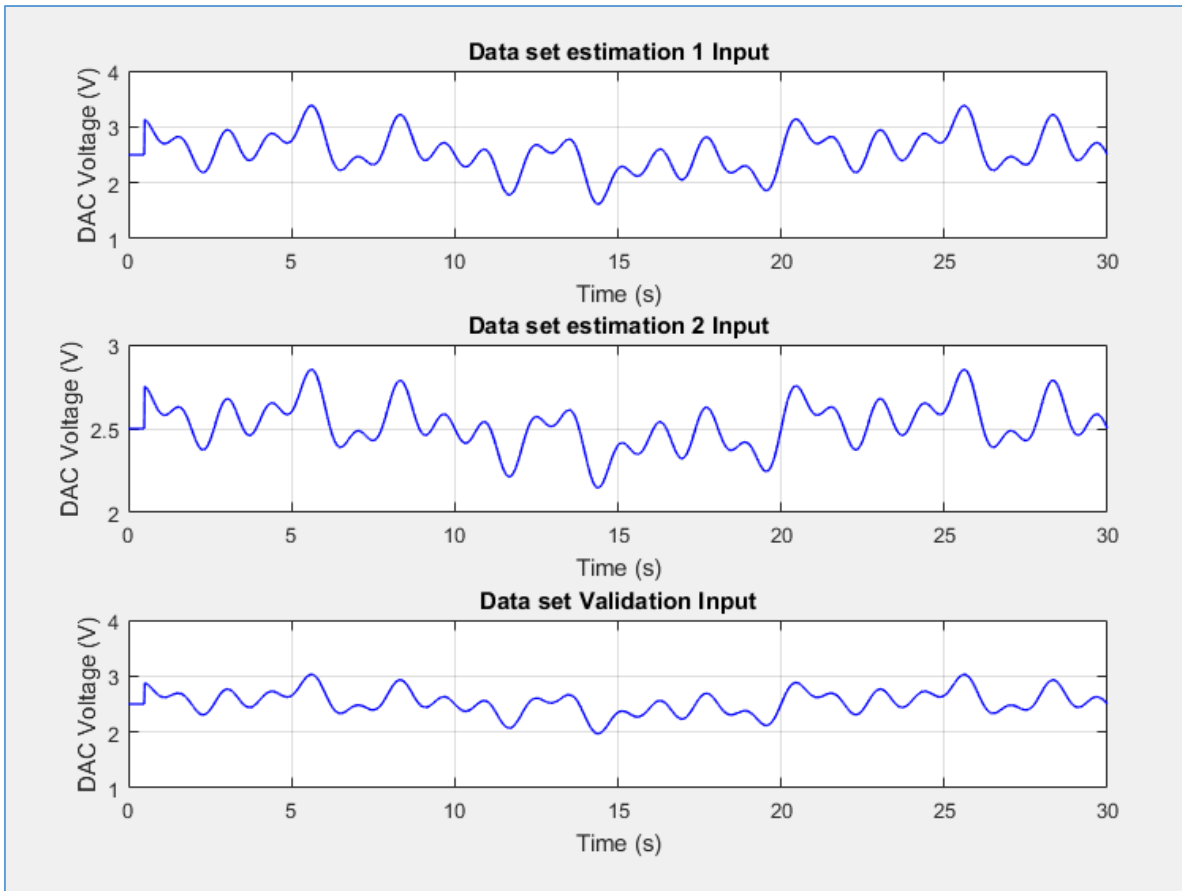


Figure 103. Inputs used for both directions movement and friction identification.

The experiments are carried out in voltage control, so the parameters identified will differ from the ones estimated with the same procedure but for current control, here the initial values are shown, the values obtained in previous steps are used as initial values, and as long as we know the values of K and D are not precise, these will remain fixed as the value calculated in the preceding step.

Modelling, identification and control of an elastic joint.

Table 31. Identification friction initial parameters and its boundaries.

Parameter\Set values	x0	x0 lower	x0 upper
JmRK	0.0073	0.00036543	0.9
Je	0.3	0.01	1
N	70	70	70
BmRK	0.299278	0.249278	0.349278
D	0.05	0.045	0.075
Be	0.1	Eps(0)	0.7
Kin	14.4	14.025	14.975
K1	40	37	43
τ_c RK	0.315	0.28	2.75
K3	150	149	151
Fs	1.60667	1.36	10
Alpha	0.465822	0.04	1.5
Beta	0.0445634	0.004	1.05

Once the model for this step is integrated in the IDNLGREY script the following results are obtained.

Table 32. Final estimated parameters for both velocity 2 filter data sets.

Search method = 'lm', 'lm'		
Parameter\Value	xf	Std. dev σ
JmRK	0.00557276	9.12065e-06
Je	0.0264691	0.000865728
N	70	0
BmRK	0.30773	0
D	0.075	0.00252879
Be	0.178733	0.00570538
Kin	14.9589	0
K1	39.8319	1.26379
τ_c RK	0.57484	3.21678e-05
K3	149.999	2292.04
Fs	1.36	0.0217614
Alpha	0.29442	0.00188563
Beta	0.27002	0.0044257

nlgr2 =

Continuous-time nonlinear grey-box model defined by 'model1' (MEX-file):

$$\frac{dx}{dt} = F(t, u(t), x(t), p1, \dots, p13)$$

$$y(t) = H(t, u(t), x(t), p1, \dots, p13) + e(t)$$

with 1 input, 4 states, 4 outputs, and 10 free parameters (out of 13).

Input:

u(1) DAC voltage(t) [V]

States: initial value

x(1) theta(t) [rad] xinit@exp1 0 (fix) in [-Inf, Inf]

x(2) q(t) [rad] xinit@exp1 -0 (fix) in [-Inf, Inf]

x(3) th_p(t) [rad/s] xinit@exp1 0.0384371 (fix) in [-Inf, Inf]

x(4) q_p(t) [rad/s] xinit@exp1 6.2353e-13 (fix) in [-Inf, Inf]

Outputs:

y(1) theta(t) [rad]

y(2) q(t) [rad]

y(3) th_p(t) [rad/s]

y(4) q_p(t) [rad/s]

Parameters:

		value	standard dev	
p1	Motor inertia [kgm ²]	0.00557276	9.12065e-06	(est) in [0.00036543, 0.9]
p2	Link side inertia [kgm ²]	0.0264691	0.000865728	(est) in [0.01, 1]
p3	HarmonicDrive reduction ratio	70	0	(fix) in [70, 70]
p4	Motor viscous friction [Nms/rad]	0.30773	0	(fix) in [0.249278, 0.349278]
p5	Spring Damping [Nms/rad]	0.075	0.00252879	(est) in [0.045, 0.075]
p6	Link viscous friction [Nms/rad]	0.178733	0.00570538	(est) in]0, 0.7]
p7	Input gain [Nm/A]	14.9589	0	(fix) in [14.025, 14.975]
p8	Spring stiffness constant [Nm/rad]	39.8319	1.26379	(est) in [37, 43]
p9	Coulomb friction	0.57484	3.21678e-05	(est) in [0.28, 2.75]
p10	k3 [k3]	149.999		(est) in [149, 151]
p11	striebeck	1.36	0.0217614	(est) in [1.36, 10]
p12	alpha	0.294424	0.00188563	(est) in [0.04, 1.5]
p13	beta	0.270028	0.00442576	(est) in [0.004, 1.05]

Name: Respuesta simulada
Status:
Termination condition: Maximum number of iterations reached.
Number of iterations: 15, Number of function evaluations: 76
Estimated using Solver: ode45; Search: lm on time domain data "Datos del experimento".
Fit to estimation data: [99.27;98.61;94.18;19.22]%
FPE: 1.331e-08, MSE: 0.9208
More information in model's "Report" property.

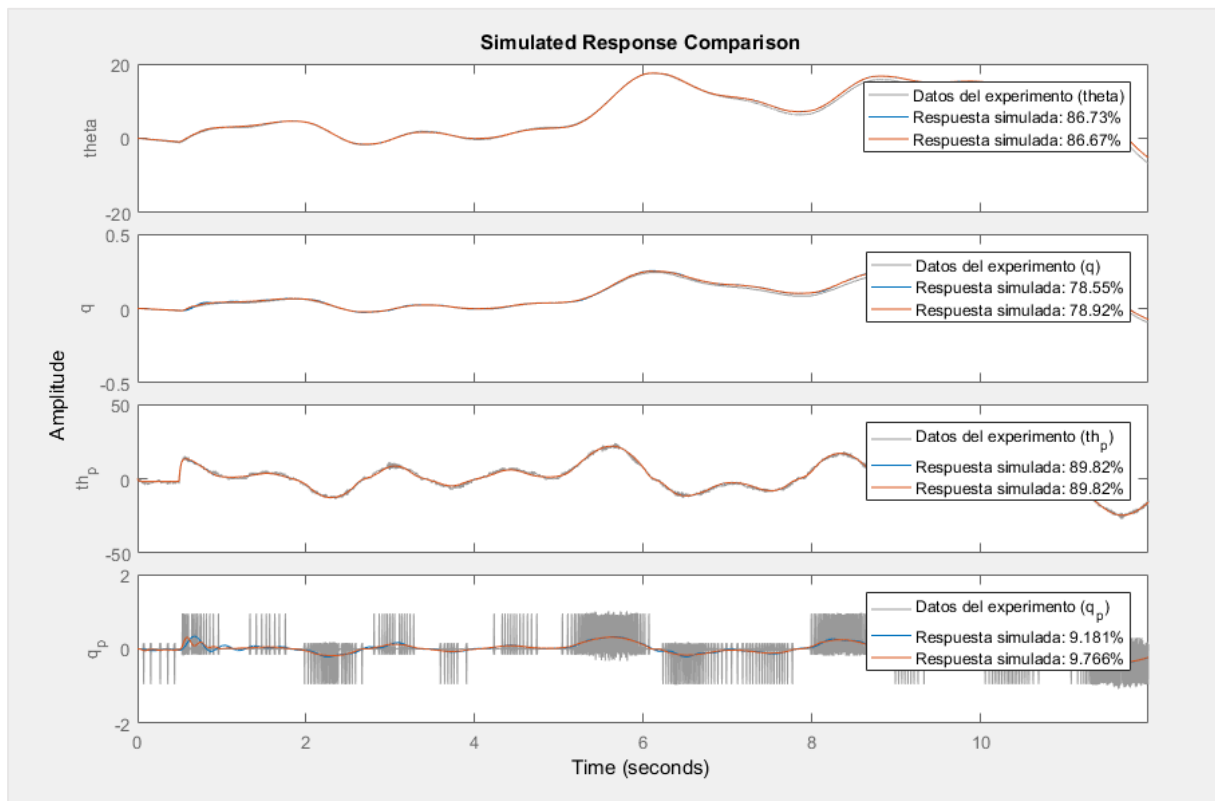


Figure 104. Simulated response comparison for the model obtained and the validation data set.

From the simulated response we can compare it with the one shown in figure 102, now the error in the position that appeared before when the velocity was crossing the zero value is gone, also remark that the standard deviations obtained are very small, this indicates that the model, a low standard deviation means that the data points are close to the expected value, while the higher value means that the data points are spread out over a wider range of values. This is logical since the only value big is related to the K which identification is highly dependent on the velocity of the link signal, that in this case isn't accurate.

Also the prediction error and the residuals are analyzed and shown in the following figures.

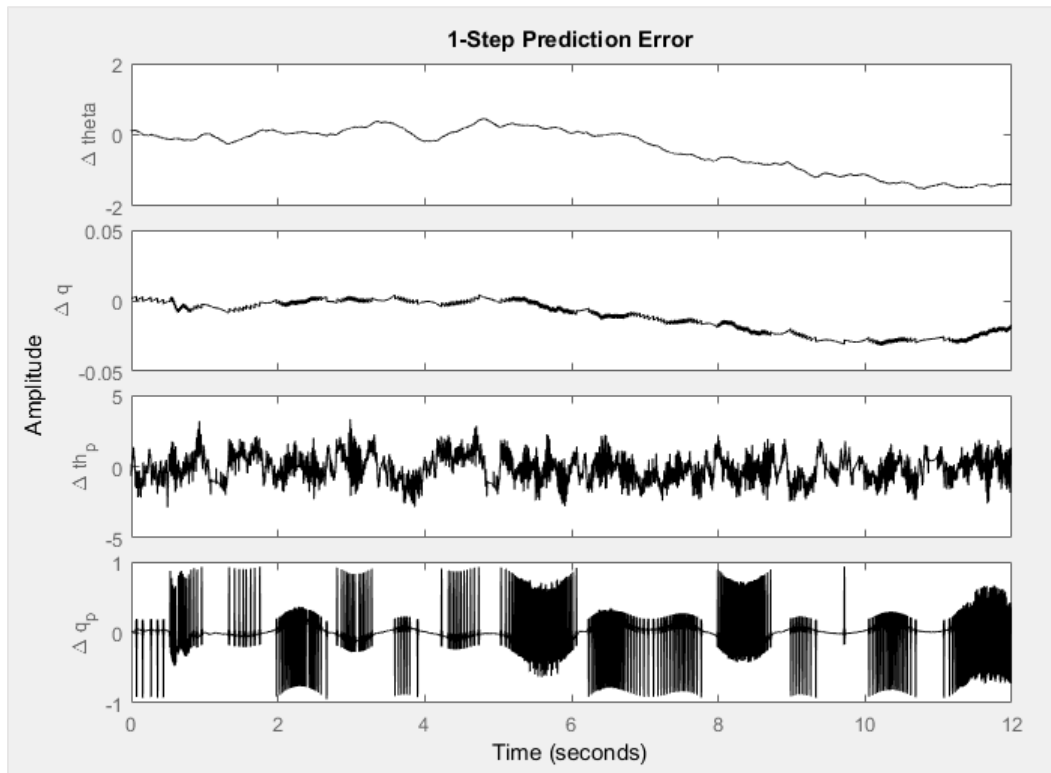


Figure 105. 1-Step prediction error for the identified model.

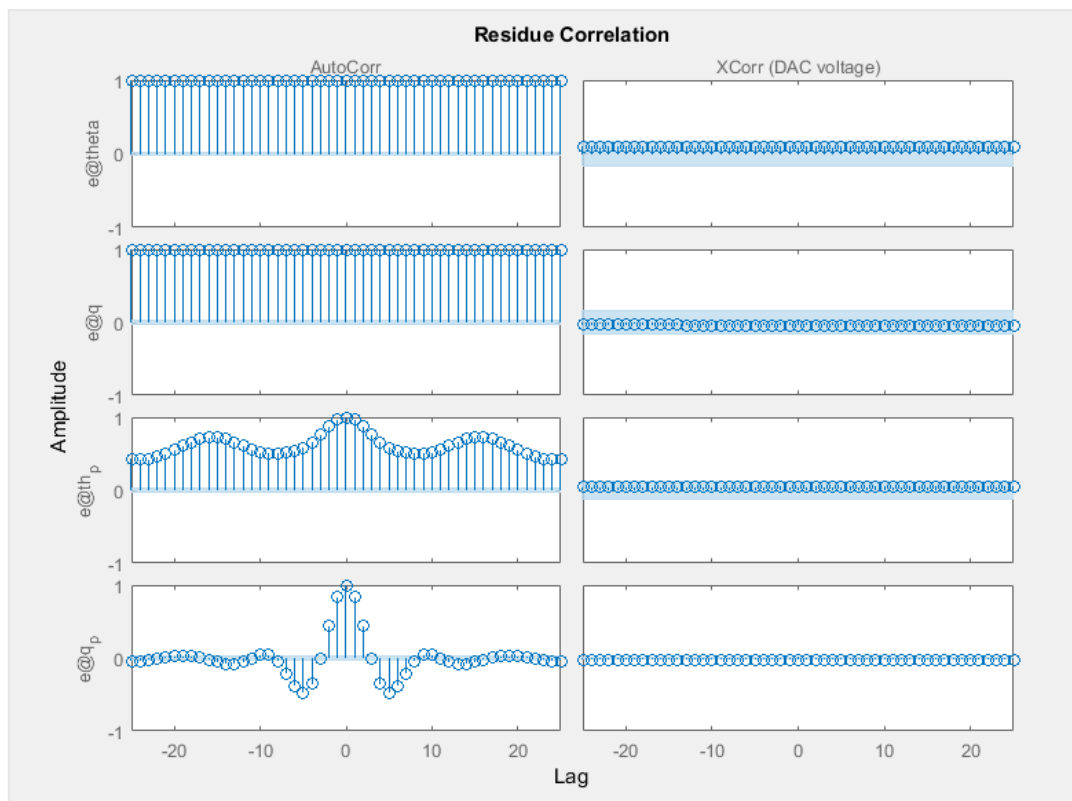


Figure 106. Residue correlation for the estimated model.

When comparing the obtained prediction error and the residuals we find that for the prediction error there is still a non modelled phenomena, this is probably the hysteresis in the kinematic error that affects the position, for the residue analysis we can conclude that the autocorrelation for both velocity signals is not ideal, as long as both are estimations from the discretized position signals, having noise and error (also induced by the kinematic error in the Harmonic Drive). Despite that, the cross correlation for the four outputs related to the input now are inside the confidence range, this means that the model now represents the reality better than previous simplified models.

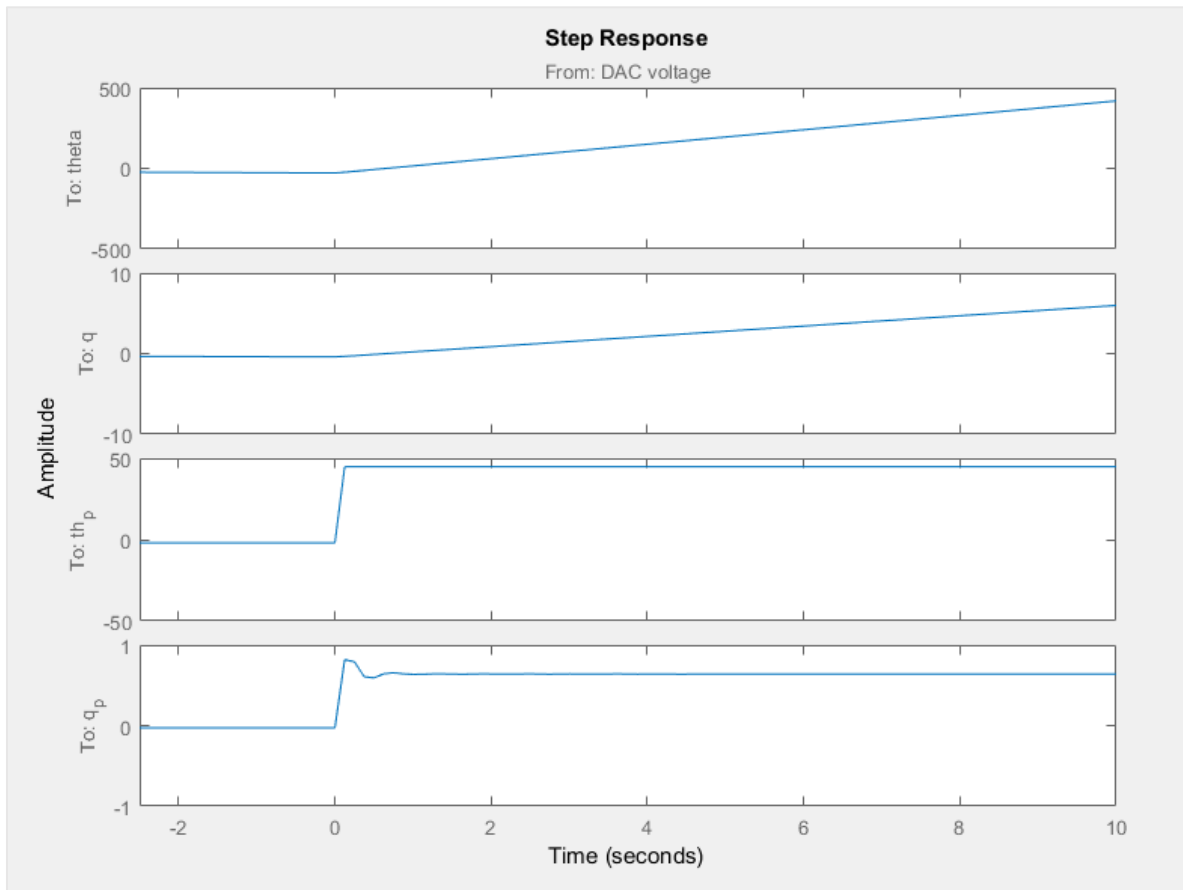


Figure 107. Step response for the identified model.

Finally the step response is analyzed, here the main problem appears in the velocity link step response, where the oscillations due to the torsional spring that is the Harmonic drive transmission are notified, in the motor side its not that obvious since the torque is reduced seventy times, so the relationship between the K and the D will affect this step response oscillations in the link side velocity, this will be analyzed later on through simulations of the system.

5.4 Current control identification

In this section the same procedure carried in 5.3.4 will be used, but for current control. In this mode it has been comprobated that for a high input in the DAC, the current cant follow the frequency, this means the ESCON acts as a low pass filter, rejecting higher frequencies, that's the main reason why these experiments have been carried with low frequencies at the input, an example of this problem is shown in the figure 109.

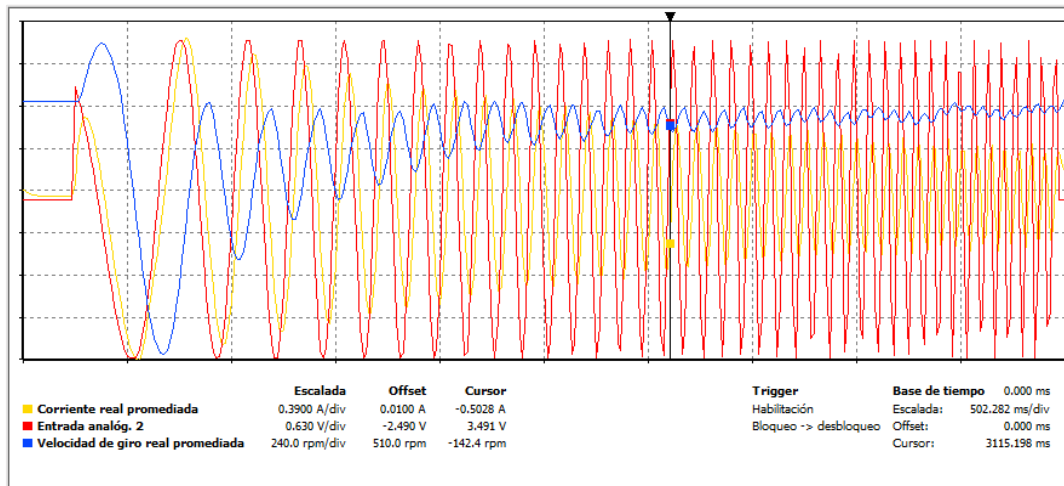


Figure 108. Current response for high frequency DAC input.

The yellow line represents the current, the red the DAC value and the blue the velocity of the motor.

5.4.1 Harmonic drive with nonlinear friction and elasticity parameter identification

So for this experiment, the following inputs are used to generate the data sets 1 and 2.

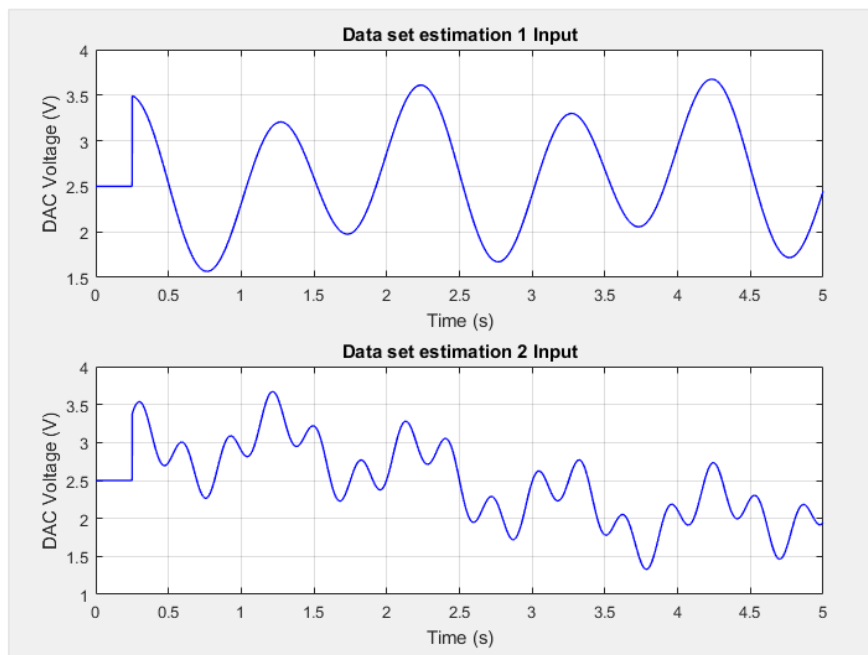


Figure 109. Low frequency input for current control identification.

Modelling, identification and control of an elastic joint.

The initial parameters as well as its boundaries are shown in the following table

Table 33. Identification friction initial parameters and its boundaries.

Parameter\Set values	x0	x0 lower	x0 upper
Jm	6.8874e-4	3.06e-6	1
Je	0.0215	0	10
N	70	70	70
Bm	0.0064	0	100
D	0.0562	1e-06	1000
Be	0.1538	Eps(0)	2
Kin	0.3090	0.002	1000
K1	40.43	2	90000
τ_c	0.0297	0.0001	0.4
alpha	0.398629	0.0004	1.75
Fs	0.2	0.0001	200
Alpha	0.399435	1e-05	200
K3	150	140	151

Once the model for this step is integrated in the IDNLGREY script the following results are obtained.

Table 34. Final estimated parameters for both velocity 2 filter data sets.

Search method = 'lm', 'lm'		
Parameter\Value	xf	Std. dev σ
Jm	0.0264089	0.160579
Je	1.34695e-06	3.21398e-06
N	70	0
Bm	0.0064	0
D	0.0562	0
Be	0.063377	0.310624
Kin	11.0924	67.191
K1	40.4364	0
τ_c	0.355817	0
K3	150	0
Fs	1.75	14.1792
Alpha	0.0001	0.260249
Beta	150	0

The estimated model properties are shown below:

```

nlgr2 =
Continuous-time nonlinear grey-box model defined by 'model_current' (MEX-file):
    dx/dt = F(t, u(t), x(t), p1, ..., p13)
    y(t) = H(t, u(t), x(t), p1, ..., p13) + e(t)
with 1 input, 4 states, 4 outputs, and 7 free parameters (out of 13).
Input:
    u(1)  DAC(t) [V]
States:
           initial value
x(1)  theta(t) [rad]  xinit@exp1           0  (fix) in [-Inf, Inf]
x(2)  q(t) [rad]     xinit@exp1          -0  (fix) in [-Inf, Inf]
x(3)  th_p(t) [rad/s] xinit@exp1    8.65465e-29  (fix) in [-Inf, Inf]
x(4)  q_p(t) [rad/s] xinit@exp1   -1.06372e-33 (fix) in [-Inf, Inf]
Outputs:
y(1)  theta(t) [rad]
y(2)  q(t) [rad]
y(3)  th_p(t) [rad/s]
y(4)  q_p(t) [rad/s]
Parameters:
           value           standard dev
p1  Motor side inertia [kgm^2]           0.0264089      0.160579  (est)
in [3.06e-06, 1]
p2  Link side inertia [kgm^2]           1.34695e-06    3.21398e-06  (est)
in [0, 10]
p3  HarmonicDrive reduction ratio           70             0  (fix)
in [70, 70]
p4  Motor viscous friction [Nms/rad]       0.0064         0  (fix)
in [0, 100]
p5  Spring Damping [Nms/rad]             0.0562         0  (fix)
in [1e-06, 1000]
p6  Link viscous friction [Nms/rad]       0.063377       0.310624  (est)
in [0, 2]
p7  Input constant [Nm/A]                11.0924        67.191  (est)
in [0.002, 1000]
p8  Spring stiffness constant [Nm/rad]     40.4364        0  (fix)
in [2, 90000]

```

p9	Coulomb friction [Nm/rad]	0.355817	0 (fix) in [0.0001, 0.4]
p10	Striebeck	1.75	14.1792 (est) in [0.0004, 1.75]
p11	Alpha_f	0.0001	1.2262 (est) in [0.0001, 200]
p12	Beta_f	0.079195	0.260249 (est) in [1e-05, 200]
p13	k3	150	0 (fix) in [140, 151]

Name: Respuesta simulada

Status:

Termination condition: Maximum number of iterations reached.

Number of iterations: 10, Number of function evaluations: 54

Estimated using Solver: ode45; Search: lm on time domain data "Datos del experimento".

Fit to estimation data: [90.31;88.01;93.16;73.2]%

FPE: 0.1007, MSE: 116

More information in model's "Report" property.

Take into account that, as long as in this step the power stage, ESCON module 50/5 is working in current control, the parameters directly identified will differ from the ones obtained in voltage control, more precisely the ones related to the trigonometric friction and those from the motor side equations.

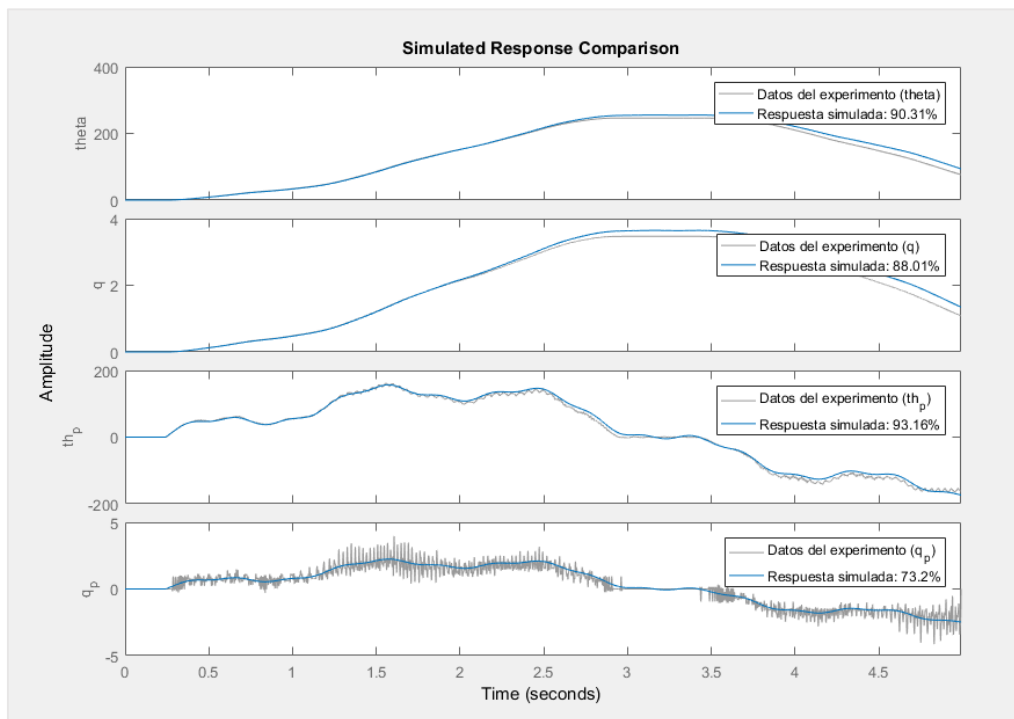


Figure 110. Simulated response comparison for the first data set.

Modelling, identification and control of an elastic joint.

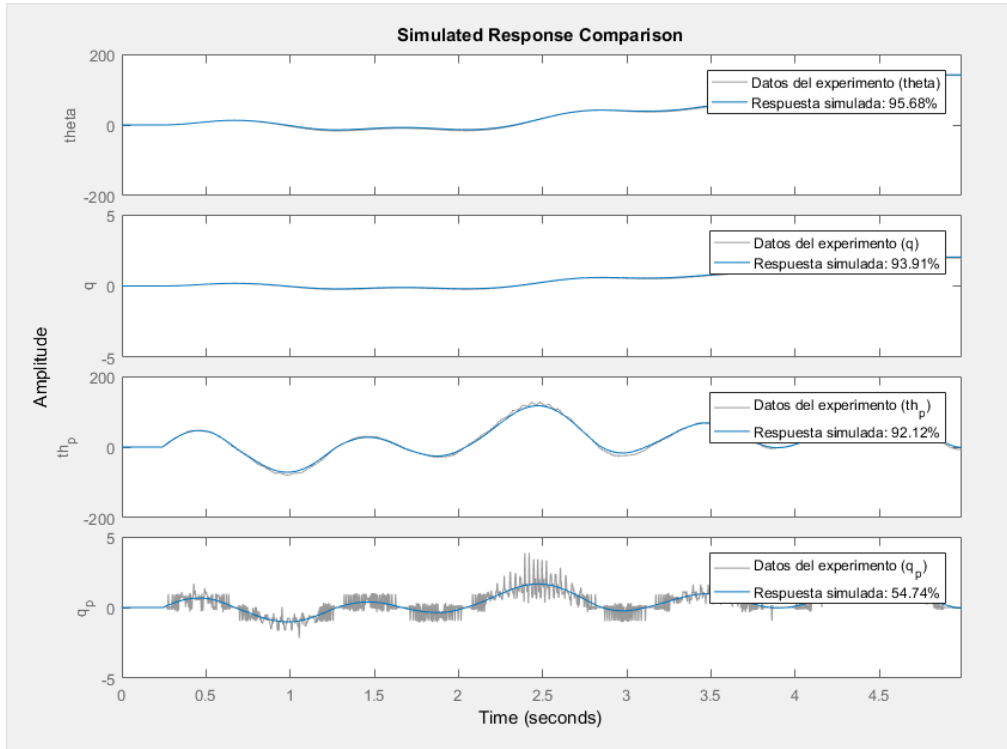


Figure 111. Simulated response comparison for the second data set.

As a result, the estimated model provides a good fit to data, now the 1-Step prediction error will be analyzed.

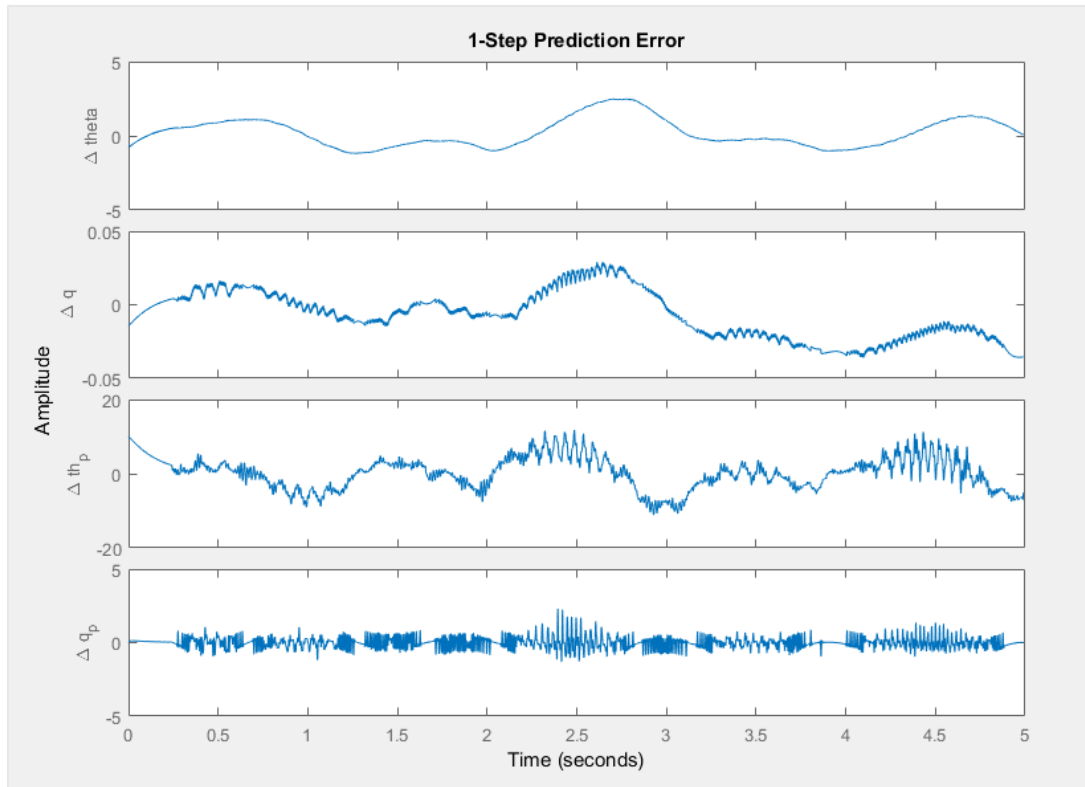


Figure 112. 1-Step prediction error.

When analyzing the 1-step error, there are some things left unexplained, as long as the ESCON is working in current control, the current is now the input to the system, this means that the real input should be the current instead the DAC voltage value programmed in the Keil code, this means that there is a noise behaviour left unexplained when considering the input used for the system.

Also the velocity oscillations due to the BLDC motor phase conmutation represented as ripples are observed, this behaviour is presented in [26], where the back EMF may take trapezoidal or sinusoidal shape. Also the error in the position is appreciated, this is mainly due to the Kinematic error, also the link velocity as long as its estimated from the link position measurements with low resolution, the quantization error appears as pulses when differentiating the position in order to obtain the velocity signal, this is also appreciated in the 1-Step prediction error.

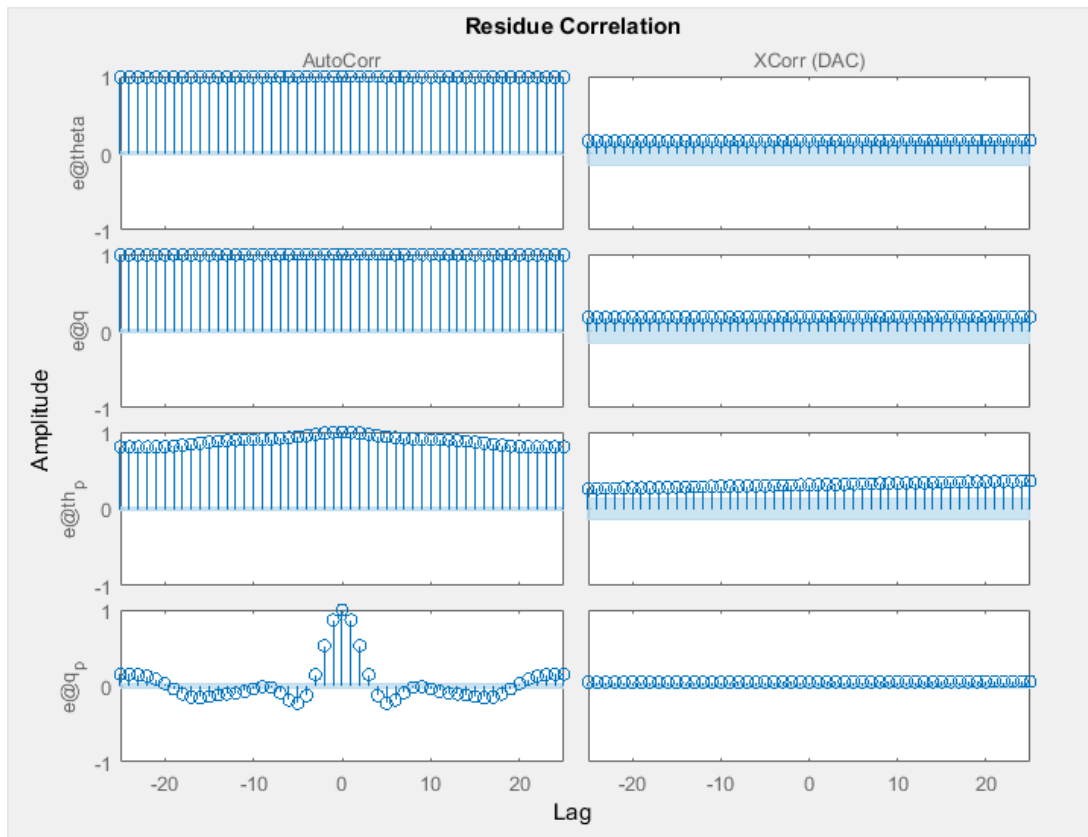


Figure 113. Residue correlation.

The residuals for the link velocity auto correlation are the same as for voltage control identifications, as well as the autocorrelations of the positions, the cross correlations are close to the confidence boundaries, this means the model can be still improved, probably with torque measurements as well as with higher resolution encoders.

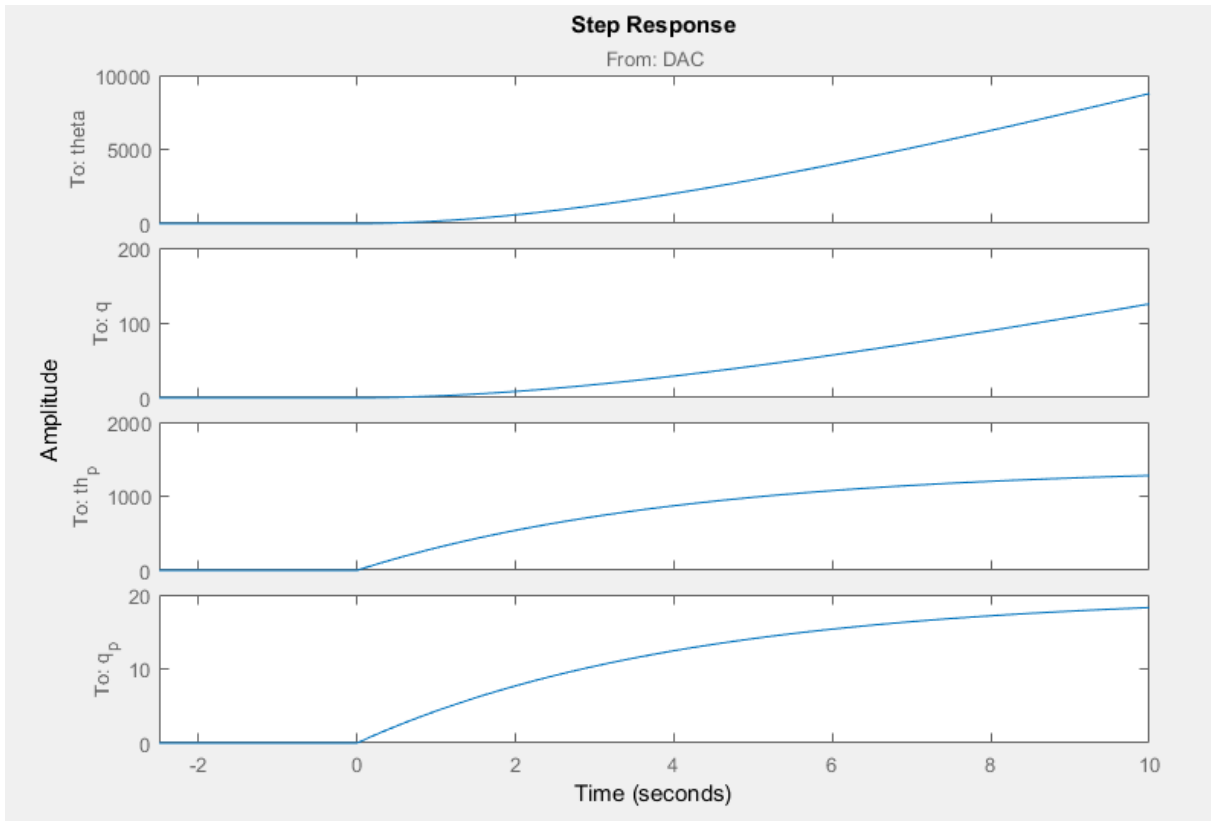


Figure 114. Step response of the identified system.

The identified step response of the system corresponds to the typical shape of a DC motor actuated by current, where there is a time until the motor reaches steady state velocity.

5.5 Frequency response analysis

When analyzing the frequency response, two transfer functions are of main interest, motor velocity respect to the current, and link velocity respect to the current.

From the equations of the system presented in section 3, (3.13) and (3.14), applying Laplace transform to the differential equations to make them algebraic expressions, first applied to the motor equation:

$$u * KaKm = J_m s^2 \theta + b_m s \theta + \frac{K}{N^2} \theta - \frac{K}{N} q + \frac{B}{N^2} s \theta - \frac{B}{N} s q \quad (5.32)$$

$$u * KaKm = \theta (J_m s^2 + (b_m + \frac{B}{N^2})s + \frac{K}{N^2}) - q (\frac{K + Bs}{N}) \quad (5.33)$$

Now obtain the motor position from link equation:

$$q (J_e s^2 + (B + b_e)s + K) = \theta (\frac{K + Bs}{N}) \quad (5.34)$$

$$q = \theta \frac{(Bs + K)}{(J_e s^2 + (B + b_e)s + K)N} ; \theta = q \frac{(J_e s^2 + (B + b_e)s + K)N}{(Bs + K)} \quad (5.35)$$

After that in the motor equation, depending on the transfer function desired (from input to motor, and from input to link) use q or θ to obtain the desired transfer function:

From input to link:

$$u * KaKm = q \left[\frac{(J_e s^2 + (B + b_e)s + K)N}{(Bs + K)} * (J_m s^2 + (b_m + \frac{B}{N^2})s + \frac{K}{N^2}) \right] - q (\frac{K + Bs}{N}) \quad (5.36)$$

$$\frac{(b_m B + b_m b_e + \frac{B^2}{N^2} + \frac{B b_e}{N^2})s^2 + (\frac{K}{N^2}(b_e + B) + K(b_m + \frac{B}{N^2})s + \frac{K^2}{N^2})}{(Bs + K)} q - (\frac{K + Bs}{N})q \quad (5.37)$$

Now rearrange with common denominator:

From input to motor:

$$u * KaKm = \theta (J_m s^2 + (b_m + \frac{B}{N^2})s + \frac{K}{N^2}) - \theta \left(\frac{(Bs + K)}{(J_e s^2 + (B + b_e)s + K)N} \right) \quad (5.38)$$

The desired TF is obtained by getting $G_{th_N} = \theta/u$, so it can be expressed as:

$$\frac{\theta}{u} = \frac{KaKm * (J_e s^2 + (B + b_e)s + K)N}{[(J_m s^2 + (b_m + \frac{B}{N^2})s + \frac{K}{N^2}) * (J_e s^2 + (B + b_e)s + K)N] - (Bs + K)} \quad (5.39)$$

Now arranging the denominator:

Modelling, identification and control of an elastic joint.

$$den_{\theta} = J_e J_m s^4 + (J_m (b_e + B) + J_e (b_m + \frac{B}{N^2})) s^3 + (K (J_m + \frac{J_e}{N^2}) + (B b_m + b_e b_m + B \frac{b_e}{N^2})) s^2 + (K (b_m + \frac{b_e}{N^2})) s \quad (5.40)$$

So after that the final expression can be analyzed for all the frequencies range:

$$G_{\theta} = \frac{\theta(s)}{u(s)} = \frac{KaKm * (J_e s^2 + (B + b_e)s + K)N}{den(s)} \quad (5.41)$$

This procedure can be repeated to obtain both transfer functions, from input to motor velocity and from input to link velocity, after that the numerators and denominators of the transfer function will be.

$$num_q = KaKm(Bs + K) \quad (5.42)$$

$$num_{\theta} = KaKm(J_e s^2 + (b_e + B)s + K) \quad (5.43)$$

$$den_q = N(J_e J_m s^4 + (J_m (b_e + B) + J_e (b_m + \frac{B}{N^2})) s^3 + (K (J_m + \frac{J_e}{N^2}) + (B b_m + b_e b_m + B \frac{b_e}{N^2})) s^2 + (K (b_m + \frac{b_e}{N^2})) s) \quad (5.44)$$

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -K/N^2 J_m & -(b_m + B/N^2)/J_m & K/N J_m & B/N J_m \\ 0 & 0 & 0 & 1 \\ K/N J_e & B/N J_e & -K/J_e & -(b_e + B)/J_e \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ 1/J_m \\ 0 \\ 0 \end{bmatrix} \tau \quad (5.45)$$

When implemented on Matlab:

```
%% Bode plot with Harmonic drive reduction
numthN = [KaKm*Je; KaKm*(B+Be); KaKm*K]';
numqN = [KaKm*B; KaKm*K]';
denNth = [Je*Jm;
          Jm*(Be+B)+Je*(Bm+B/(N^2));
          K*(Jm+Je/(N^2))+(B*Bm+Bm*Be+(B*Be/(N^2)));
          K*(Bm+Be/(N^2));
          0]';
denNq = N*[Je*Jm;
           Jm*(Be+B)+Je*(Bm+B/(N^2));
           K*(Jm+Je/(N^2))+(B*Bm+Bm*Be+(B*Be/(N^2)));
           K*(Bm+Be/(N^2));
           0]';
GthN=tf(numthN,denNth);GqN=tf(numqN,denNq);
bode(GthN,GqN)
```

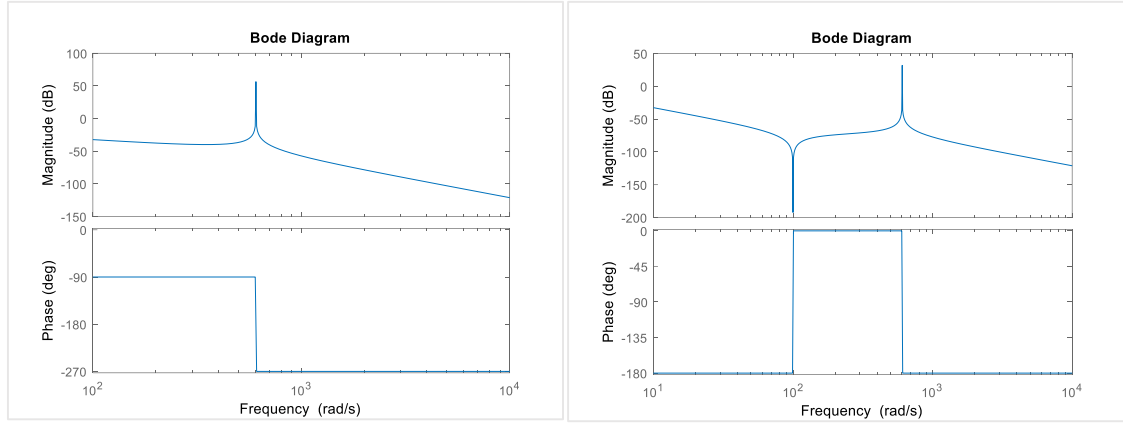



Figure 115. Typical bode plot shape of link velocity respect to the current(left), and motor velocity respect to the current (right)

As stated in [10], the relation between torque and motor velocity can be written as:

$$den_{\theta} = s^3 J_e J_m + s^2 \left(J_e B_m + D \left(J_m + \frac{J_e}{N^2} \right) \right) + s \left(K \left(J_m + \frac{J_e}{N^2} \right) + D B_m \right) + K B_m \quad (5.46)$$

If the effect of the damping coefficient, D , is negligible the transfer function becomes:

$$num_{\theta} = J_e s^2 + K \quad (5.47)$$

$$den_{\theta} = s^3 J_e J_m + s^2 J_e B_m + s K \left(J_m + \frac{J_e}{N^2} \right) + K B_m \quad (5.48)$$

The parameter B_m affects the low frequency region, for this reason, the value estimated of this parameter can be considered good, the fact that this parameter has been identified with different procedures with the same result also is a good reason to validate this parameter value.

The variation of the coefficient D , does not make any evident changes in the Bode plot, this means D is hard to estimate, as it can be seen in the results, this parameter give many different values for different identification procedures.

The parameters K and J_e have similar but opposite effects on the Bode plot, the angular frequency of the notch in the Bode plot of the physically parametrized model corresponds to the zero in the previous equation and it can be calculated as:

$$\omega_{n1} = Im \left\{ -\frac{D}{2J_e} \pm \sqrt{\left(\frac{D}{2J_e}\right)^2 - \frac{K}{J_e}} \right\} \approx \sqrt{\frac{K}{J_e}} \quad (5.49)$$

The stiffer the spring is the higher the frequency will be, the depth of the notch is only dependent on $\frac{D}{2J_e}$, the smaller this values is, the deeper the notch becomes.

6. Simulations and trajectory generator

6.1 Simulations with the parameters identified

Simulation with obtained parameters, with the same input as the DAC signal stored in the variable “tension_s” from workspace, the Harmonic drive model in the simulations will be evaluated in different scenarios, depending on the considered physical phenomenas in the model.

For the parameters which value has the uncertainty of being well identified, mainly corresponding the elastic torque related parameters K, D the values will be varied in the simulation and its effects will be evaluated, this is also influent in the hysteresis modelling as long as the signal that provides most of the information about this effects are the link side signals, this is derived from the fact that the elastic torque is reduced by $N = 70$ in the motor side and so its effect in the motor side signals is not noticeable.

6.1.1 Model simulation

In this section the model identified in voltage control is implemented in simulink, where the values of the parameters can be changed in order to see how these affect to the response of the system, in future works, more complex models can be implemented or compared with the ones obtained.

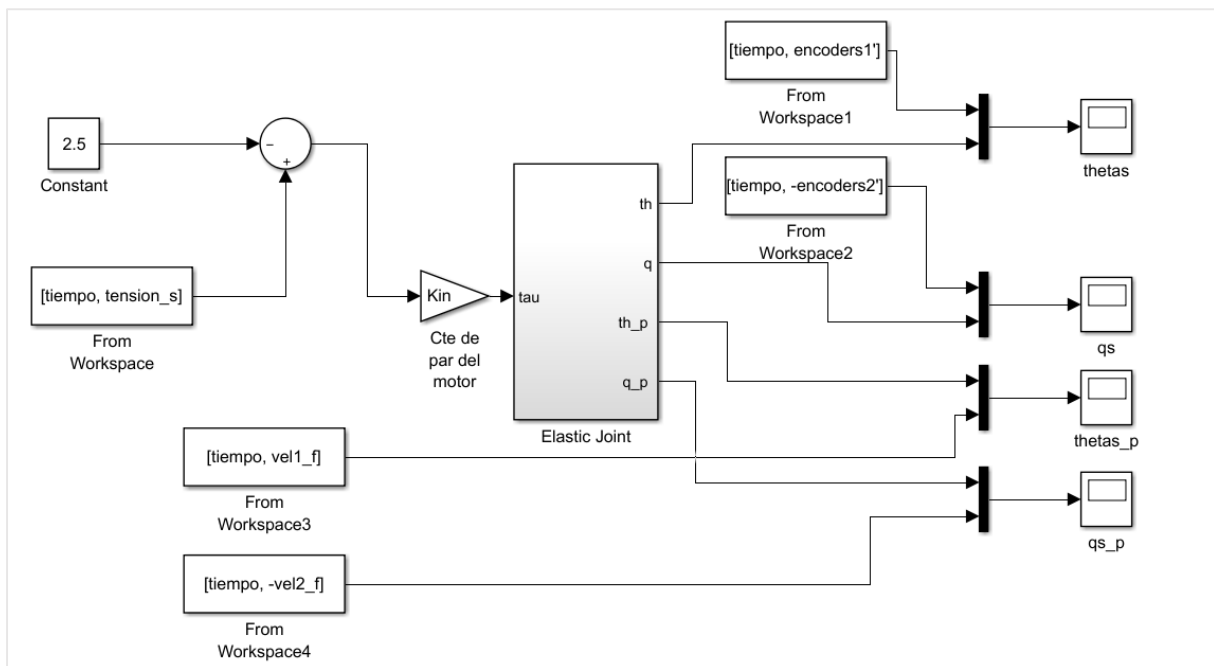


Figure 116. General view of the elastic joint model implemented in Simulink.

Modelling, identification and control of an elastic joint.

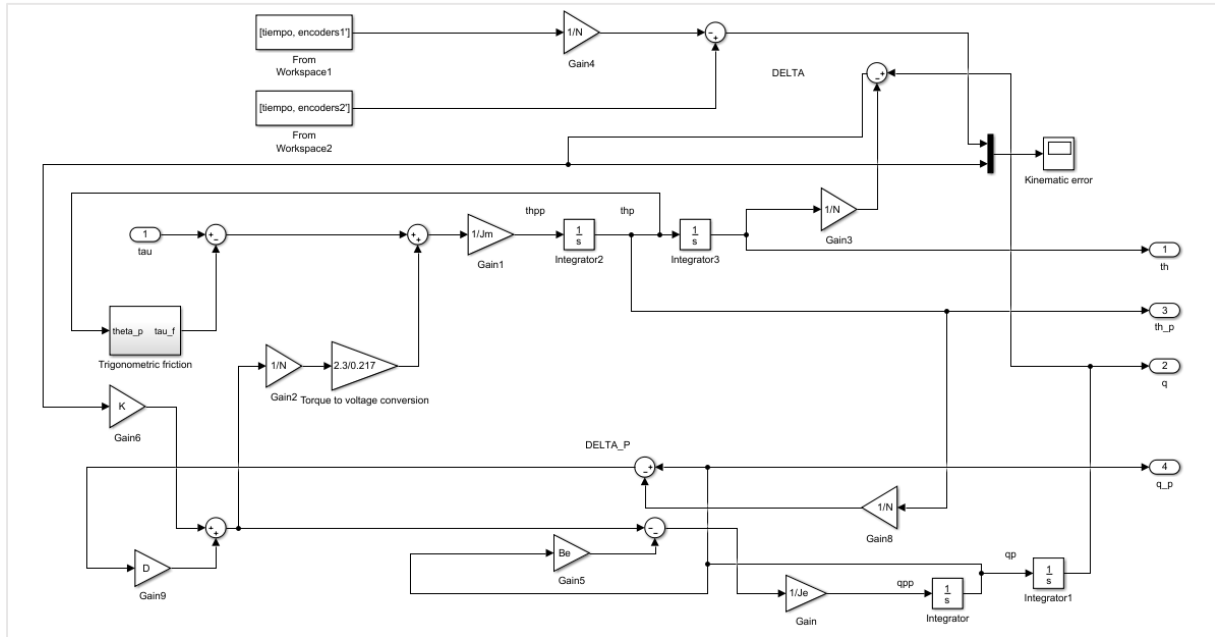


Figure 117. Elastic joint Simulink implementation.

In this Simulink implementation, the real data for any experiment can be introduced and compared to the signals, this might help also in the future for taking a good initial guess for the parameters in the identification procedure.

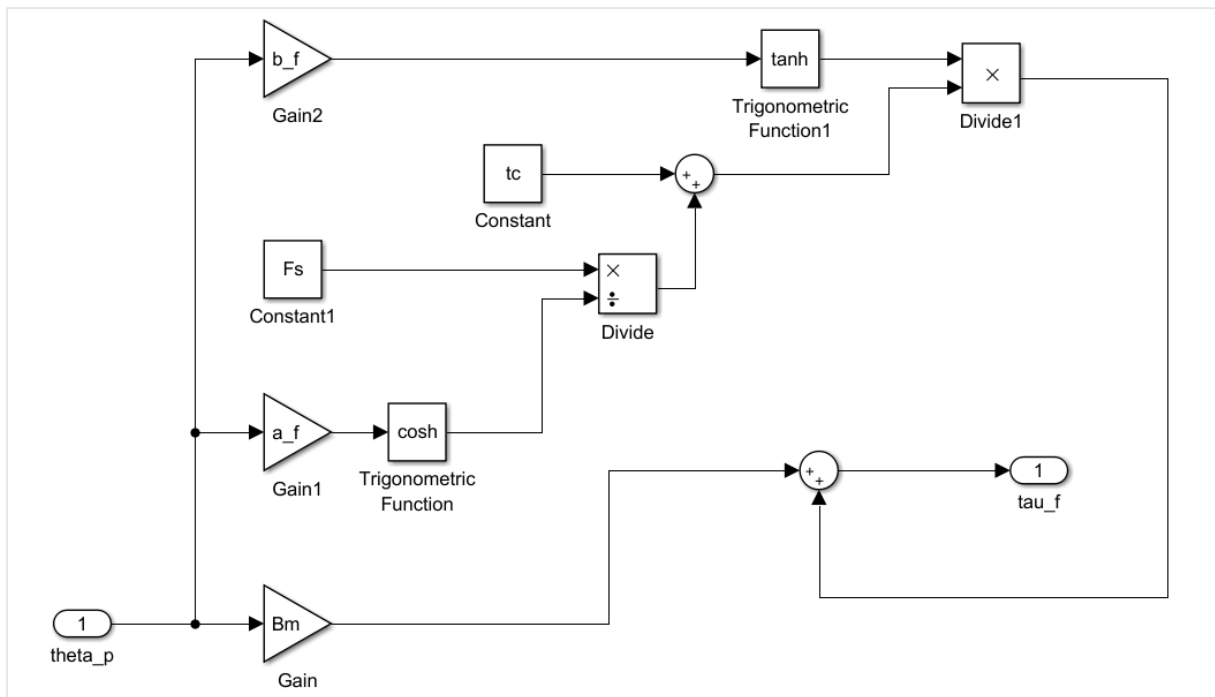


Figure 118. Trigonometric function for the friction implemented on Simulink.

Here is shown as an example how the real data is compared to values close to the ones identified.

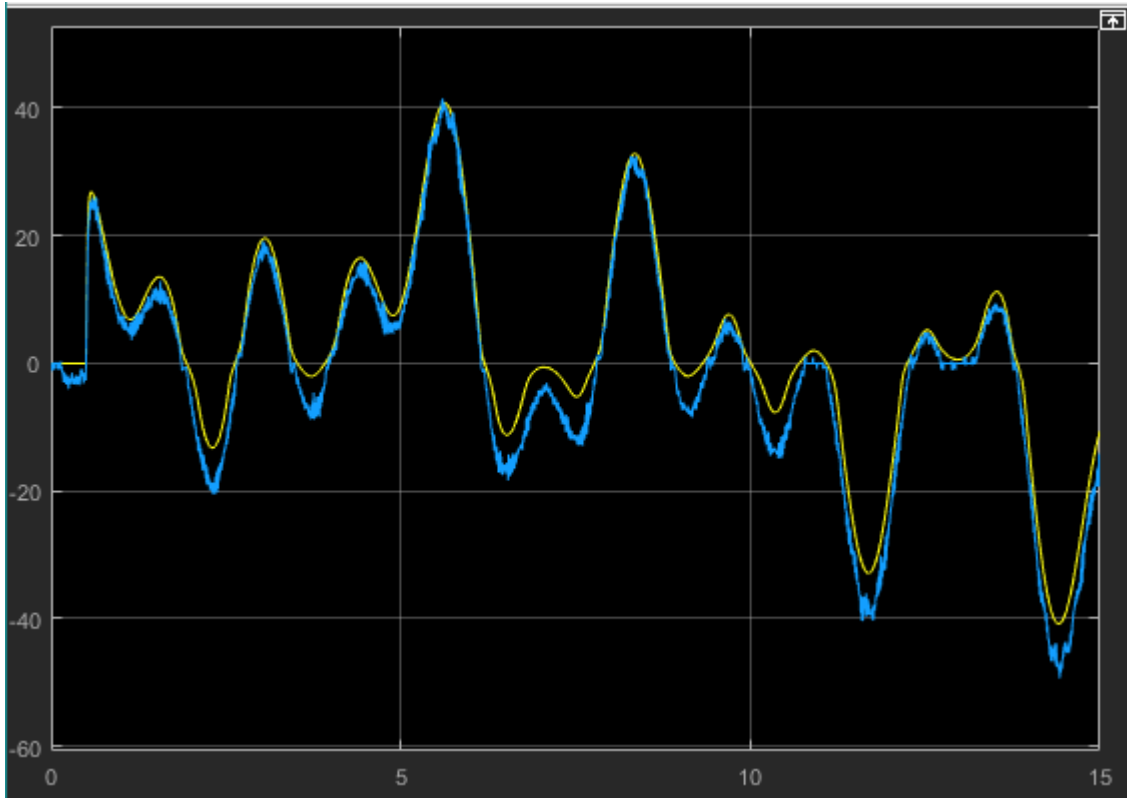


Figure 119. Simulink real data vs simulated signal.

Modelling, identification and control of an elastic joint.

6.1.2 Bouc-Wen model

In order to see how the Bouc-Wen model behaves it has been implemented in simulink, in a simple inertial system actuated, the general view of this model is shown in the figure below.

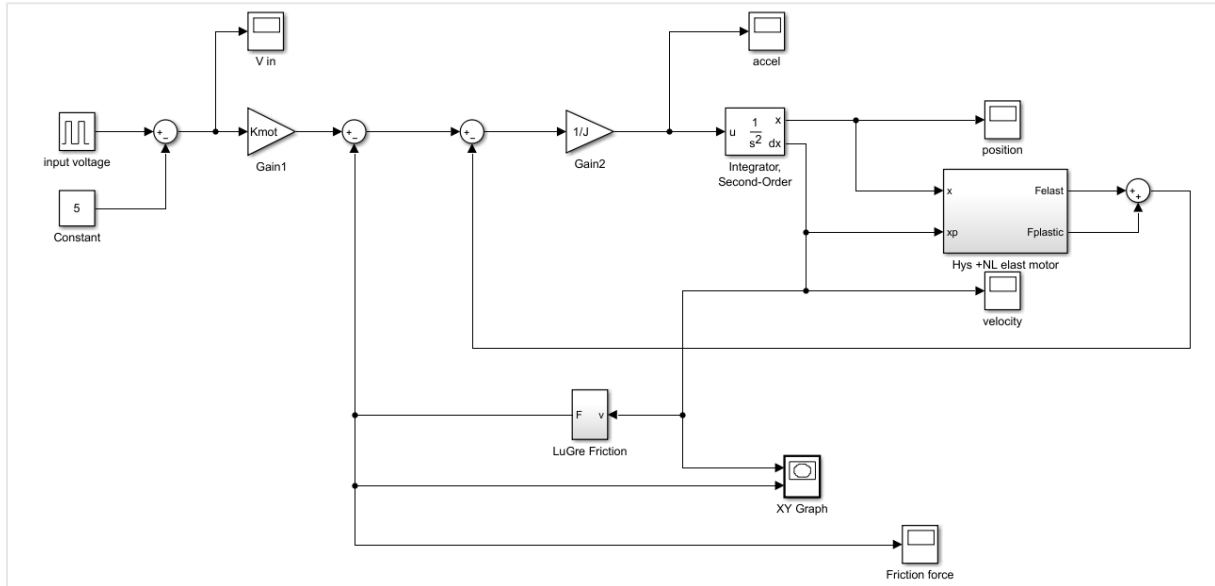


Figure 120. Bouc-Wen model implementation in simulink.

And the hysteresis plus nonlinear cubic polynomial for the elasticity is implemented in the subsystem, where the internal state as well as the balance between purely elastic and purely plastic behaviour is implemented.

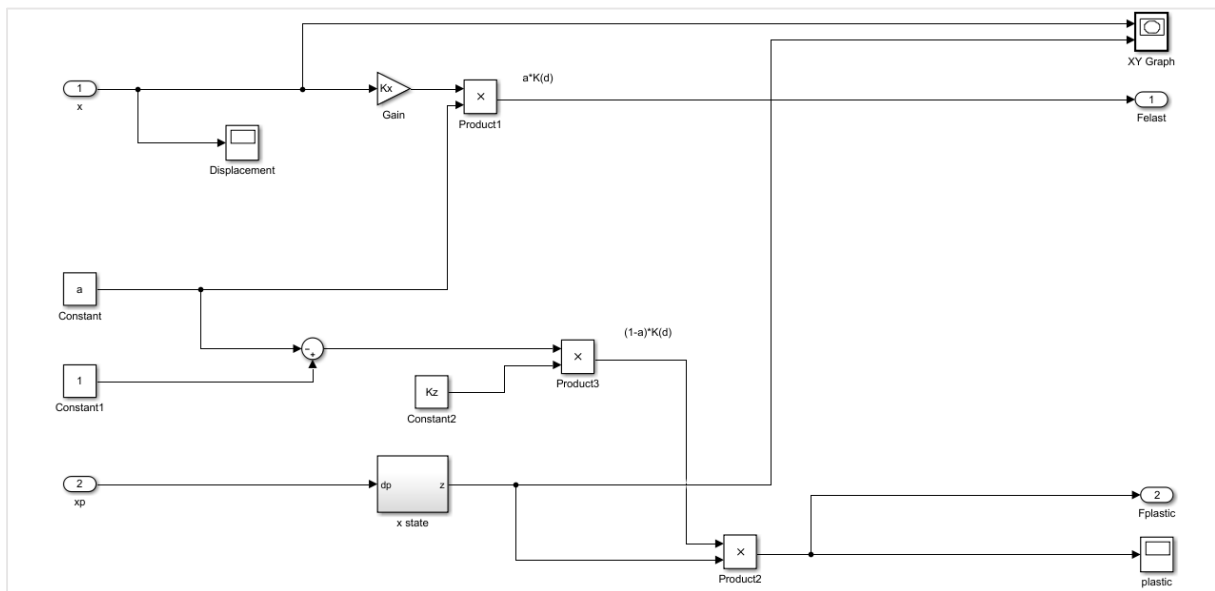


Figure 121. Bouc-Wen subsystem for plastic-elastic balance.

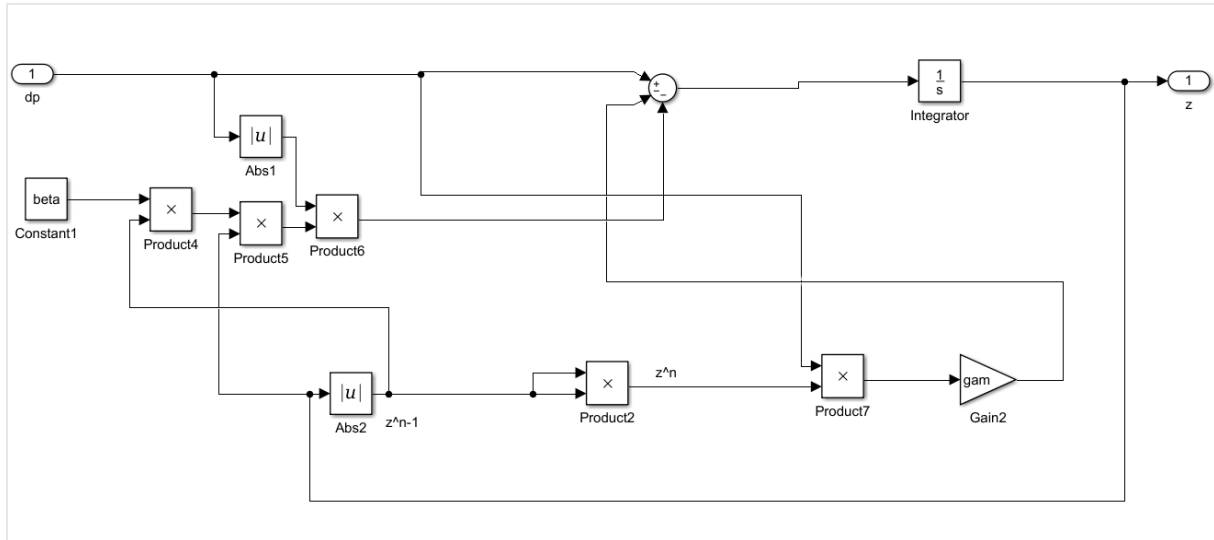


Figure 122. Bouc-Wen internal state Simulink implementation.

As it can be seen in the general view of the model, also the LuGre model has been implemented as a friction actuating in the motor. When simulating the following parameters were chosen for simulation, then the plots of the velocity against the friction force will be plot in a XY graph, in order to see the typical LuGre shape, then the same plot is done for observing the hysteresis between the internal state and the displacement, where a memory effect will be appreciated, and the typical hysteresis shape will be plot.

```

%% Bouc Wen hysteresis
mi = 0.5;
a = mi;
A = 1;
beta=0.8;
gam=0.7;
Kz=1;
Kx=10;

%% Striebeck function
Ms = 1; % Fricción Striebeck
Mc = 0.8; % Fricción Coulomb.
vs = 0.092; %Velocidad Striebeck.

%% Mechanical system with friction

J = 1; % inertia moment
Kmot = 0.5; % motor constant

%% LuGre Parameters
sig2 = 0.5;
sig0= 100;
sig1=1.70;
    
```

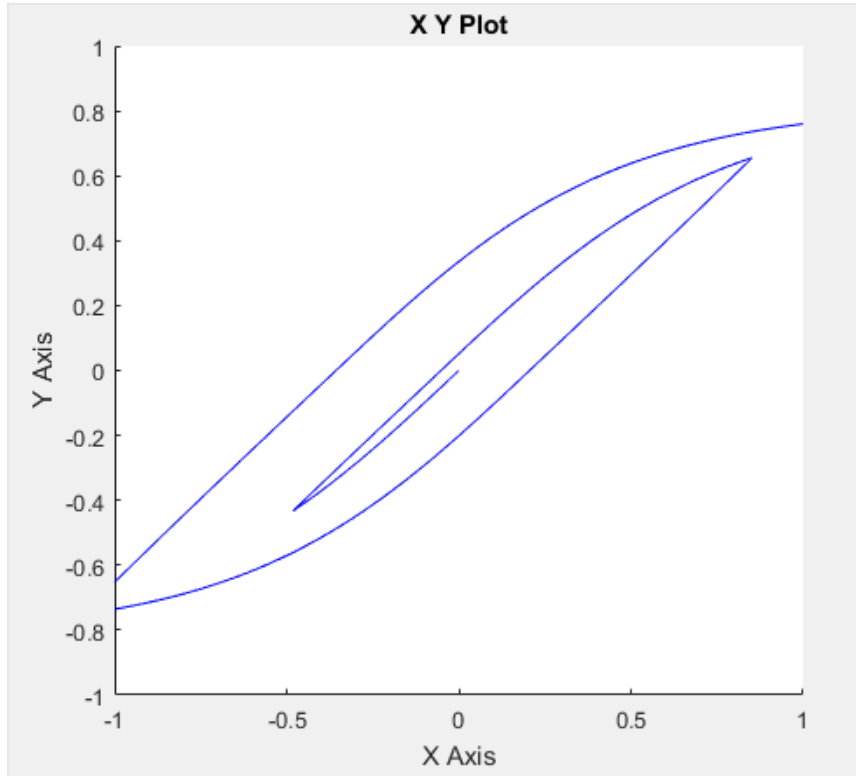


Figure 123. Hysteresys shape obtained by simulation.

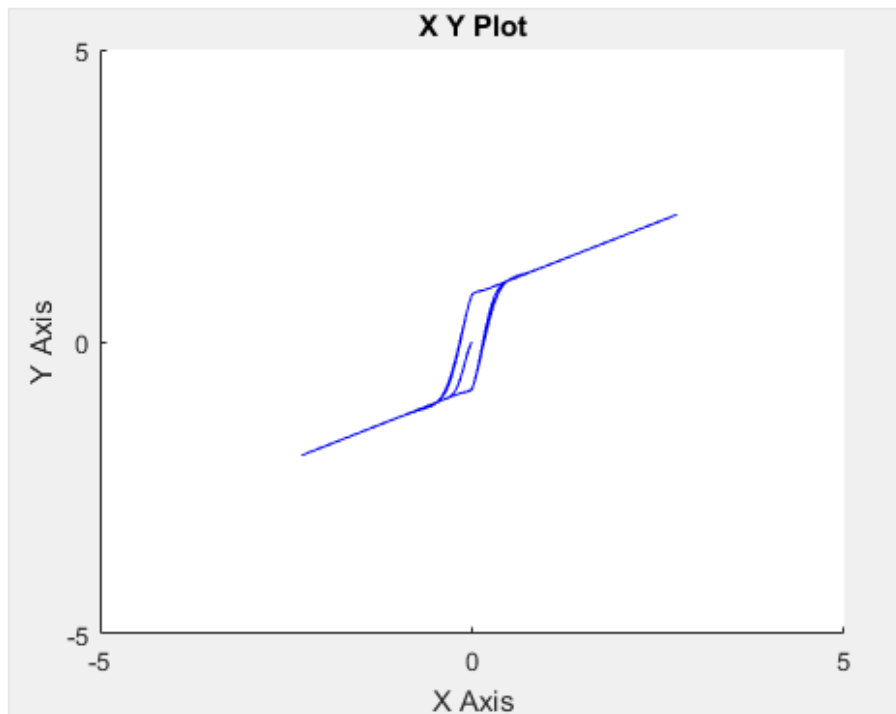


Figure 124. Friction force vs velocity, x axis velocity, y axis friction force.

6.2 Fourth order trajectory generator for Keil uVision

A fourth order trajectory generator was implemented for future trajectory tracking tasks, the function speed_signal generates the 4th order trajectory with smooth acceleration transitions.

It takes as an argument the time and the maximum value, the signal generated is symmetric and the middle part is constant, so it only returns half the signal.

To use it properly, the function has to be called 2 times. The first one to generate the signal until half of the experiment time, then call the function with the time as a parameter until $t_s/2$. The second call is for the symmetric part of the signal, calling the function with $(t_{\text{experiment}} - t)$ as a parameter will generate the symmetric part. The signal can be modified by changing the value of $t_1, t_2 \dots$ and V_{max}

```
#include "math.h"
extern volatile float v_max;
extern volatile float final_pos;
float speed_signal(float i, float v_max, float final_pos){
    float signal;
    float k=3.19870022581016*v_max*v_max*v_max*v_max/(final_pos*final_pos*final_pos);
    float laps = 0.0909521553765987*final_pos/v_max;
    float t1 = laps;
    float t2 = 2*laps;
    float t3 = 3*laps;
    float t4 = 4*laps;
    float t5 = 5*laps;
    float t6 = 6*laps;
    float t7 = 7*laps;
    float t8 = 11*laps;
    float t9 = 12*laps;
    float t10 = 13*laps;
    float t11 = 14*laps;
    float t12 = 15*laps;
    float t13 = 16*laps;
    float t14 = 17*laps;
    float t15 = 18*laps;
```



```
if(i <= t1)
signal = k/6*i*i*i;
if(i>t1 && i<= t2)
signal = (k*t1*(i*i/2-t1*i/2) + k*t1*t1*t1/6);
if(i>t2 && i<=t3)
signal = (k*((t1+t2)*i*i/2 - i*i*i/6 - (t1*t1/2+t2*t2/2)*i)
+ k*(t2*t2*t2/6 + t1*t1*t1/6));
if(i>t3 && i<=t4)
signal = ((k*((t1+t2)*t3 - t3*t3/2) - k*(t1*t1/2 + t2*t2/2) )*i
+ (k*(t1*t1*t1 - 3*t1*t3*t3 + t2*t2*t2 - 3*t2*t3*t3 + 2*t3*t3*t3)/6));
if(i>t4 && i<=t5)
signal = (k*(t4*i*i/2 - i*i*i/6) + (k*(-t1*t1 + 2*t1*t3 - t2*t2 + 2*t2*t3 - t3*t3 - t4*t4)/2)*i
+ k*(t1*t1*t1 - 3*t1*t3*t3 + t2*t2*t2 - 3*t2*t3*t3 + 2*t3*t3*t3 + t4*t4*t4)/6);
if(i>t5 && i<=t6)
signal = (k*(t4-t5)*i*i/2 + k*(-t1*t1 + 2*t1*t3 - t2*t2 + 2*t2*t3 - t3*t3 - t4*t4 + t5*t5)/2)*i
+ (k*(t1*t1*t1 - 3*t1*t3*t3 + t2*t2*t2 - 3*t2*t3*t3 + 2*t3*t3*t3 + t4*t4*t4 - t5*t5*t5)/6));
if(i>t6 && i<=t7)
signal = (k*(i*i*i/6 - t7*i*i/2) + (k*(-t1*t1 + 2*t1*t3 - t2*t2 + 2*t2*t3 - t3*t3 - t4*t4 + 2*t4*t6 +
t5*t5 - 2*t5*t6 - t6*t6 + 2*t6*t7)/2)*i
+ (k*(t1*t1*t1 - 3*t1*t3*t3 + t2*t2*t2 - 3*t2*t3*t3 + 2*t3*t3*t3 + t4*t4*t4 - 3*t4*t6*t6 -
t5*t5*t5 + 3*t5*t6*t6 + 2*t6*t6*t6 - 3*t6*t6*t7)/6));
if( i>t7 )
signal = (k*(t7*t7*t7/6 - t7*t7*t7/2) + (k*(-t1*t1 + 2*t1*t3 - t2*t2 + 2*t2*t3 - t3*t3 - t4*t4 +
2*t4*t6 + t5*t5 - 2*t5*t6 - t6*t6 + 2*t6*t7)/2)*t7
+ (k*(t1*t1*t1 - 3*t1*t3*t3 + t2*t2*t2 - 3*t2*t3*t3 + 2*t3*t3*t3 + t4*t4*t4 - 3*t4*t6*t6 -
t5*t5*t5 + 3*t5*t6*t6 + 2*t6*t6*t6 - 3*t6*t6*t7)/6));

return signal;
}
```

7. Conclusions and future works

After finishing the project, many concepts studied were reviewed and implemented in the experimental platform used. During the project, many problems were arising when trying to make the system work in a proper way in different areas such as:

- Electronic circuit design.
- Mechanical structure building.
- Mathematical modeling.
- Microcontroller programming.
- Signal processing.
- Identification techniques.
- Matlab programming.
- Real world limitations.

As stated in previous chapters, the results are not good enough for knowing the complete real behaviour of the system in order to apply control techniques to the real plant. Despite that with the given procedure if the experimental platform is improved, the project could be finished with more satisfactory results, this implies finishing the identification accurately for the elasticity parameters associated to the Harmonic Drive. This could be achieved by implementing load sensors and isolating the motor-reductor system into a lock-load configuration in order to carry out different experiment with torque measurements, from where accurate models of hysteresis in the elastic force can be achieved.

As long as this project is part of a bigger project as mentioned in section 1 “WRAD”. The Harmonic Drive transmissions will be implemented in the two first links of the robotic arm; this implies the identification of the whole robot arm with two elastic joints and normal transmissions for the last link and attached to the end effector.

This robot will be worn in a vest by the patient so it will be an autonomous system, this means that the robot will have batteries and that power consumption is a crucial fact. For saving power, as much as possible optimal control would be applied in order to carry a control strategy that balances between energy consumption and trajectory tracking.

For the prototype implementation, many Discovery boards will be needed as well as a communication protocol between them, the before mentioned control would be implemented in a Raspberry, which is perfect for matrix operations programmed in python.

8. Budget

The project was carried in an experimental platform with material that have been used in previous projects; anyway, an estimation of the value of all the materials used is presented.

Table 35. Material budget.

Name of the component	Price per unit(€)	Units
Maxon BLDC Motor	130	1
Discovery STM32F429Zi	31.21	1
Power Supply PULS Dimension Q-Series	229	1
ESCON Module 50/5	139.18	1
ESCON Power cable	14.82	1
ESCON Motor cable	25.91	1
ESCON Hall sensor cable	14.64	1
USB Mini cable	1.75	2
USB Standard cable	6.95	1
Encoder header HEDX-9141	24.82	2
Codewheels	11.12	2
Encoder 3 channels cable	9.80	2
Operational Amplifier	0.60	1
Resistances	0.1	4
Microcontroller cables	0.25	10

9. References

- [1] Timothy Douglas Tuttle. "Understanding and Modeling the Behavior of a Harmonic Drive Gear Transmission". *Massachusetts Institute of Technology (MIT) Artificial Intelligence Laboratory*, 1992.
- [2] Vincent Lampaert, Farid Al-Bender, Jan Swevers. "A Generalized Maxwell-Slip Friction Model appropriate for Control Purposes". *PhysCon*, 2003.
- [3] Michel Ruderman, Torsten Bertram, Makoto Iwasaki. "Modeling, observation, and control of hysteresis torsion in elastic robot joints". *Mechatronics* 24 407-415, 2014.
- [4] W. Seyfferth, A.J. Maghzal, J. Angeles. "Nonlinear modeling and parameter identification of Harmonic Drive robotic transmissions". *IEEE International conference on robotics and automation, 1999. 0-7803-1965-6/95*.
- [5] Fathi H. Ghorbel, Prasanna S. Gandhi, Friedhelm Alpetter. "On the kinematic error in harmonic drive gears". *Transactions of the ASME, Vol. 123, March 2001*.
- [6] Michel Ruderman, "Modeling of elastic robot joints with nonlinear damping and hysteresis." *DOI: 10.5772/25494, January 2011*.
- [7] Bruno Siciliano, Springer handbook of robotics, chapter 13. Robots with flexible elements.
- [8] Ovidiu Solomon, "Some typical shapes of hysteretic loops using the Bouc-Wen model."
- [9] Cristina Castejón, Juan Carlos García-Prada, Omar José Lara, "Modelado de una transmisión Harmonic drive. Análisis del error cinemático." *Revista iberoamericana de ingeniería mecánica, Vol. 13 Nº2 pp 51-65, 2009*.
- [10] Stig Moberg, "On modeling and control of flexible manipulators." *Linköping studies in science and technology. Thesis No 1336*.
- [11] H.D. Taghirad, P.R. Bélanger, "Modelling and parameter identification of Harmonic drive systems." *Journal of Dynamic systems, measurements and control, ASME publications*.
- [12] Mohammed Ismail, Fayçal Ikhouane, José Rodellar. "The hysteresis Bouc-Wen model, a survey" *Arch Comput Methods Eng (2009) 16: 161-188, DOI 10.1007/s11831-009-9031-8*.
- [13] T. Tjahjowidodo, F. Al-Bender, H. Van Brussel. "Nonlinear modelling and identification of torsional behaviour in Harmonic drives." *Proceedings of ISMA, 2006*.
- [14] H. Olsson, K.J. Aström, C. Canudas de Wit, M. Gäfvert, P. Lischinsky. "Friction models and friction compensation." *European journal of control, 176-195. 1998*.
- [15] Dr. Eleni Chatzi, K. Agathos, G. Abbiati. "Method of finite elements II, modeling of Hysteresis." *Institute of structural engineering (IBK), DBAUG, ETH Zurich, Structural mechanics slides*.
- [16] *Miniature gearheads, Harmonic drive gearhead, CSF Mini series, datasheet*.
- [17] Demosthenis D. Rizos, Spilios D. Fassois, "Friction identification based upon the LuGre and Maxwell slip models." *Interntional federation of automatic and control, 2005*.
- [18] T. Tjahjowidodo, F. Al-Bender, H. Van Brussel. "Friction identification and compensation in a DC motor." *Interntional federation of automatic and control, 16th Triennial world congress, 2005*.

- [19] Junki Oaki, Shuichi Adachi, "Grey box modeling of elastic joint robot with Harmonic drive and timing belt." *Interntional federation of automatic and control, 16th IFAC Symposium on system identification, July 2012.*
- [20] Erik Wernholt, Svante Gunnarsson, "Nonlinear grey-box identification of industrial robots containing flexibilities." *Interntional federation of automatic and control, 16th Triennial world congress, 2005.*
- [21] Hamid D. Taghirad, "On the modeling and identification of Harmonic drive systems." *Centre for intelligent machines, McGill University, CIM-TR-97-02, January 1997.*
- [22] "Nonlinear least squares data fitting." Appendix D.
- [23] David di Ruscio, "Prediction error methods", *Telemark University College, revised April 3, 2014.*
- [24] A. Croeze, L. Pittman, W. Reynolds, "Solving nonlinear least-squares problems with the Gauss-Newton and Levenberg-Marquardt methods."
- [25] Henri P. Gavin, "The Levenberg-Marquardt method for nonlinear least squares curve fitting problems."
- [26] C.Cham, Z.Bin Samad, "Brushless DC motor Electromagnetic torque estimation with single-phase current sensing."

DOCUMENT II:

ANNEXES

DOCUMENT II: ANNEXES

1.	Matlab Scripts and functions.....	1
1.1	Voltage control.....	1
1.1.1	Reduced model (Total inertia).....	1
1.1.2	Harmonic drive simplified model (Linear elasticity).....	5
1.1.3	Harmonic drive model with nonlinear elasticity and trigonometric friction.	10
1.1.4	Harmonic drive model with nonlinear elasticity, trigonometric friction and hysteresis.	15
1.2	Current control.....	21
1.2.3	Harmonic drive model with nonlinear elasticity and trigonometric friction.	21
2.	Keil uVison code	26
2.1	Main.c.....	26
2.2	Fonctions.c	32
2.3	Init.c.....	33
2.4	Encoder.c.....	39
2.5	Stm32f4xx_it.c.....	42

1. Matlab Scripts and functions

1.1 Voltage control

1.1.1 Reduced model (Total inertia)

1.1.1.1 Reduced model MEX file

```
void compute_dx(double *dx, double *x, double *u, double **p)
{
    /* Declaration of model parameters and intermediate
    variables. */
    double *JtRK, *bmRK, *KaRK, *tcRK;

    double tau_f, sign; z

    /* Retrieve model parameters. */
    JtRK = p[0]; /* Scaled total inertia */
    bmRK = p[1]; /* Scaled viscous friction */
    KaRK = p[2]; /* Input gain */
    tcRK = p[3]; /* Scaled Coulomb friction */

    tau_f = bmRK[0]*x[0];

    /* State equations. */
    /* x[0] motor speed */
    dx[0] = ((KaRK[0]*u[0]-tcRK[0])-tau_f)/JtRK[0];
}

/* Output equations. */
void compute_y(double y[], double x[])
{
    y[0] = x[0];
}
```

Annexes.

1.1.1.2 Identification script for low frequency inertias

```
%% COMPILE THE FILE CONTAINING THE MODEL
clc, clear;
addpath('C:\Users\Rafael C\Desktop\Ident_low_freq')
run ('script_3experiments_lf');
mex Reduced_mod.c
vel = vell_noise;
vel_b = vell_noise_b;
vel_c = vell_noise_c;
%% INPUT, OUTPUT AND STATE NAME AND UNITS DEFINITIONS
InputName = {'DAC voltage'};
InputUnit = {'V'};
StateName = {'th_p'};
StateUnit = {'rad/s'};
OutputName = StateName(1);
OutputUnit = StateUnit(1);
%% PARAMETERS
n_p = 4; % Number of parameters
ParName = {'Jt*R/Ka'; 'Velocity term (Kw + R/Ka*Bm)'; 'Kin'; 'tcRK'};
ParUnit = {''; '''; '''; '''};
ParValue= {0.02; 0.28; 0.217; 0.37};
ParMin= {1E-6; 0.26; 0.01; 0.27};
ParMax = {5; 0.3; 100; 0.47};
%% INITIAL STATES OF THE EXPERIMENT
val_state = 252;
%InitialStates Data set 1
InitialStates = {vel(val_state)};
InitialStates = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates, ...
'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);
%InitialStates Data set 2
InitialStates_b = {vel_b(val_state)};
InitialStates_b = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates_b, ...
'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);
```

Annexes.

```
% Create the idnlgrey object and the iddata object
FileName      = 'Reduced_mod';           % File describing the model
structure.

Order         = [1 1 1];                % Model orders [ny nu nx].

Parameters    = struct('Name', ParName, 'Unit', ParUnit, 'Value', ParValue,
'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);

nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, 0, ...
    'Name', 'Respuesta simulada', 'InputName', InputName, ...
    'InputUnit', InputUnit, 'OutputName', OutputName, ...
    'OutputUnit', OutputUnit, 'TimeUnit', 's');

z = iddata({vel(val_state:end), vel_b(val_state:end), vel_c(val_state:end)},
{tension_s(val_state:end)-2.5          tension_s_b(val_state:end)-2.5
tension_s_c(val_state:end)-2.5}, 0.001);

z.Name = 'Datos del experimento';
z.InputName = nlgr.InputName;
z.InputUnit = nlgr.InputUnit;
z.OutputName = nlgr.OutputName;
z.OutputUnit = nlgr.OutputUnit;
z.ExperimentName = {'Estimation' 'Validation1' 'Validation2'};
z.Tstart = 0;
z.TimeUnit = 's';
present(z)

% Identification of the parameters with the model structure specified in
the idnlgrey object
% First data set: Estimation
opt = nlgreyestOptions('SearchMethod', Searchmethod, 'Display', 'on');
opt.SearchOption.MaxIter = 10;
% First data set estimation
nlgr = nlgreyest(getexp(z,1), nlgr, opt);
% Get estimated parameters in the first step
for i=1:n_p
ParValue(i)={nlgr.Parameters(i).Value};
end
% Introduce the parameters (ParValue) in the new structure
Parameters = struct('Name', ParName, 'Unit', ParUnit, 'Value', ParValue, ...
    'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates_b, Ts, ...
    'Name', 'Respuesta simulada', 'InputName', InputName, ...
    'InputUnit', InputUnit, 'OutputName', OutputName, ...
    'OutputUnit', OutputUnit, 'TimeUnit', 's');
```

Annexes.

```
%% Third data set: Validation 2
% Get estimated parameters in the second step
for i=1:n_p
ParValue(i)={nlgr.Parameters(i).Value};
end
% Introduce the parameters (ParValue) in the new structure (Parameters)
for the next estimation
Parameters = struct('Name', ParName, 'Unit', ParUnit, 'Value', ParValue,
...
    'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);
% Build the idnlgrey object with the parameters estimated in the last
step and
% with the new initial states
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates_c, Ts, ...
    'Name', 'Respuesta simulada', 'InputName', InputName, ...
    'InputUnit', InputUnit, 'OutputName', OutputName, ...
    'OutputUnit', OutputUnit, 'TimeUnit', 's');
% Estimate again with the second data set and the parameters obtained in
the previous step.
opt = nlgreyestOptions('SearchMethod', Searchmethod, 'Display', 'on');
opt.SearchOption.MaxIter = 10;
nlgr = nlgreyest(getexp(z,3), nlgr, opt);
%% Plot data and compare to identified NLGR and parameter result
evaluation
%% Compare the estimated model response with the 3 data sets
n_ds = 3; % Number of data sets
figure
compare(getexp(z,n_ds), nlgr);
%% 1 step prediction error
figure
pe(getexp(z,n_ds), nlgr);
%% Check residuals
figure('Name',[nlgr.Name ': residuals of estimated model']);
resid(getexp(z,n_ds),nlgr);
%% Check step response and covariance
figure
step(nlgr);
getcov(nlgr)
nlgr.NoiseVariance
present(nlgr);
```

Annexes.

1.1.2 Harmonic drive simplified model (Linear elasticity)

1.1.2.1 Simplified H.D model (Linear elasticity, no hysteresis, simple friction)

```
void compute_dx(double *dx, double *x, double *u, double **p){
    /* Declaration of model parameters and intermediate variables. */
    double *JmKR, *Jje, *N, *D, *be, *bmKw, *Kin, *k1, *tcKR;
    double delta_e, delta_ep, tau_e, tau_f;
    JmKR = p[0];    /* Momento de inercia del motor
    Jje = p[1];    /* Momento de inercia del elemento
    N = p[2];      /* Factor de reduccion del HarmonicDrive
    bmKw=p[3];
    D=p[4];
    be=p[5];
    Kin=p[6];
    k1 = p[7];
    tcKR = p[8];
    delta_e = x[1]-x[0]/N[0];    // Torsional angle
    delta_ep = x[3]-x[2]/N[0];   // Torsional speed
    tau_e = k1[0]*delta_e+D[0]*delta_ep;    // Linear elastic torque
    tau_f = bmKw[0]*x[2]; //Coulomb + viscous friction
    /* State equations. */
    /* x[0] angulo del motor */
    /* x[1] angulo del elemento*/
    /* x[2] velocidad del motor*/
    /* x[3] velocidad del elemento*/
    dx[0] = x[2];
    dx[1] = x[3];
    dx[2] = (Kin[0]*u[0]-tcKR[0]+(tau_e*10.5991/N[0])-tau_f)/JmKR[0];
    dx[3] = (-tau_e-be[0]*x[3])/Jje[0];
}
/* Output equations. */
void compute_y(double y[], double x[])
{
    y[0] = x[0];
    y[1] = x[1];
    y[2] = x[2];
    y[3] = x[3];}
```

Annexes.

1.1.2.2 Identification script for HD simple model

```
%% COMPILE THE FILE CONTAINING THE MODEL
clc, clear;
addpath('C:\Users\Rafael C\Desktop\Ident_high_freq')
run ('script_3experiments');
mex modell.c
%% INPUT, OUTPUT AND STATE NAME AND UNITS DEFINITIONS
InputName = {'DAC voltage'};
InputUnit = {'V'};
StateName = {'theta'; ... %Motor angle
             'q'; ... %Link angle
             'th_p'; ... %Motor speed
             'q_p'; ... %Link speed
             };
StateUnit = {'rad'; 'rad'; 'rad/s';...
            'rad/s';...
            };
OutputName = StateName(1:4);
OutputUnit = StateUnit(1:4);
%% PARAMETERS
n_p = 9; % Number of parameters
ParName = {'Motor inertia'; ...
           'Link side inertia '; ...
           'HarmonicDrive reduction ratio'; ...
           'Motor viscous friction';...
           'Spring Damping';...
           'Link viscous friction';...
           'Current to torque motor constant';...
           'Spring stiffness constant';...
           'Coulomb friction';...
           };
ParUnit = {'kgm^2'; 'kgm^2'; '' ; ...
          'Nms/rad'; 'Nms/rad'; 'Nms/rad'; 'Nm/A'; 'Nm/rad'; '' ;...
          };
Bm0 = 0.299278;
```


Annexes.

```
%PARAMETER INITIALIZATION AND BOUNDARIES
%           Jm      Jje  N    Bm  D    Be      Kin      Kl      tc
ParValue={0.00736543; 1; 70; Bm0; 0.05; 0.001; 14.4; 5000 ;0.3007 ...
};
ParMin={0.0073; 0; 70; Bm0-0.1;0.001; 0.00095; 14.325; 10;0.28;...
};
ParMax={0.00742;10; 70; Bm0+0.1; 1; 0.009; 14.475; 80000;0.315;...
};
Parameters(4).Fixed = true;
%% INITIAL STATES OF THE EXPERIMENT
%Initial state instant
val_state = 10;
vell = vell_noise;%vell_f;
vell_b = vell_noise_b;%vell_f_b;
vel2 = vel2_noise;%vel2_f;
vel2_b = vel2_noise_b;%vel2_f_b;
%InitialStates Data set 1
InitialStates = {encoders1(val_state) ; -encoders2(val_state);
vell(val_state); -vel2(val_state)};
InitialStates = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates, ...
'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);
%InitialStates Data set 2
InitialStates_b = {encoders1_b(val_state) ; -encoders2_b(val_state);
vell_b(val_state); -vel2_b(val_state)};
InitialStates_b = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates_b, ...
'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);
%% Create the idnlgrey object and the iddata object
FileName = 'modell1'; % File describing the model
structure.
Order = [4 1 4]; % Model orders [ny nu nx].
Parameters = struct('Name', ParName, 'Unit', ParUnit, 'Value',
ParValue, ...
'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, 0, ...
'Name', 'Respuesta simulada', 'InputName', InputName, ...
'InputUnit', InputUnit, 'OutputName', OutputName, ...
'OutputUnit', OutputUnit, 'TimeUnit', 's');
```

Annexes.

```
z = iddata([encoders1(val_state:end)' -encoders2(val_state:end)'
vell1(val_state:end) -vel2(val_state:end)],...
          [encoders1_b(val_state:end)' -encoders2_b(val_state:end)'
vell1_b(val_state:end) -vel2_b(val_state:end)]...
          [encoders1_c(val_state:end)' -encoders2_c(val_state:end)'
vell1_c(val_state:end) -vel2_c(val_state:end)]..
          ),{tension_s(val_state:end)-2.5 tension_s_b(val_state:end)-
2.5...
          tension_s_c(val_state:end)-2.5...
          }, 0.001);

z.Name = 'Datos del experimento';
z.InputName = nlgr.InputName;
z.InputUnit = nlgr.InputUnit;
z.OutputName = nlgr.OutputName;
z.OutputUnit = nlgr.OutputUnit;
z.ExperimentName = {'Estimation' 'Validation1' 'Validation2'};
z.Tstart = 0;
z.TimeUnit = 's';
present(z)
%% Identification of the parameters with the model structure specified
in the idnlgrey object
%% First data set: Estimation
% Estimation options (Searchmethod, max iterations...)
Searchmethod = 'lsqnonlin'; % 'lsqnonlin' 'gna' 'lm'
opt = nlgreyestOptions('SearchMethod', Searchmethod, 'Display', 'on');
opt.SearchOption.MaxIter = 5;
% First data set estimation
nlgr = nlgreyest(getexp(z,1), nlgr, opt);
%% Second data set: Validation 1
% Get estimated parameters in the first step
for i=1:n_p
ParValue(i)={nlgr.Parameters(i).Value};
end
% Introduce the parameters (ParValue) in the new structure (Parameters)
for the next estimation
Parameters = struct('Name', ParName, 'Unit', ParUnit, 'Value', ParValue,
...
'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);
```

Annexes.

```
% Fix parameters from nlgr
Parameters(4).Fixed = true;

% Build the idnlgrey object with the parameters estimated in the last
step with the new initial states
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates_b, Ts, ...
    'Name', 'Respuesta simulada', 'InputName', InputName, ...
    'InputUnit', InputUnit, 'OutputName', OutputName, ...
    'OutputUnit', OutputUnit, 'TimeUnit', 's');

% Estimate again with the second data set and the parameters obtained in
the previous step.
opt = nlgreyestOptions('SearchMethod', Searchmethod, 'Display', 'on');
opt.SearchOption.MaxIter = 30;
nlgr = nlgreyest(getexp(z,2), nlgr, opt);

% Compare the estimated model response with the 3 data sets
n_ds = 2; % Number of data sets

figure
compare(getexp(z,n_ds), nlgr);

%% 1 step prediction error
figure
pe(getexp(z,n_ds), nlgr);

%% Check residuals
figure('Name', [nlgr.Name ' : residuals of estimated model']);
resid(getexp(z,n_ds), nlgr);

%% Check step response and covariance
figure
step(nlgr);
%
% getcov(nlgr)
% nlgr.NoiseVariance

present(nlgr);
```

Annexes.

1.1.3 Harmonic drive model with nonlinear elasticity and trigonometric friction.

1.1.3.1 H.D model (Cubic elasticity, no hysteresis, trigonometric friction)

```
void compute_dx(double *dx, double *x, double *u, double **p)
{
    /* Declaration of model parameters and intermediate variables. */
    double *JmKR, *Jje, *N, *D, *be, *bmKw, *Kin, *k1, *tcKR, *k3, *Fs,
    *alpha_f, *beta_f;
    double delta_e, delta_ep, tau_h, tau_f, k_nl;
    double c_c = 2.3/0.217; // Conversion from torque to voltage with
    motor constants
    /* Retrieve model parameters. */
    JmKR = p[0]; /* Momento de inercia del motor
    Jje = p[1]; /* Momento de inercia del elemento
    N = p[2]; /* Factor de reduccion del HarmonicDrive
    bmKw=p[3];
    D=p[4];
    be=p[5];
    Kin=p[6];
    k1 = p[7];
    tcKR = p[8];
    k3 = p[9];
    Fs = p[10];
    alpha_f = p[11];
    beta_f = p[12];
    /*Ecuaciones + NL elasticidad*/
    /*Torsional angle*/
    delta_e = x[1]-x[0]/N[0]; // Torsional angle
    delta_ep = x[3]-x[2]/N[0]; // Torsional speed
    k_nl = k1[0]*delta_e+k3[0]*pow(delta_e,3);
    tau_h = k_nl+D[0]*delta_ep;
    tau_f =
    bmKw[0]*x[2]+(tcKR[0]+Fs[0]/cosh(alpha_f[0]*x[2])*tanh(beta_f[0]*x[2]))
;
    /* State equations. */
    /* x[0] angulo del motor */
    /* x[1] angulo del elemento*/
    /* x[2] velocidad del motor*/
    /* x[3] velocidad del elemento*/
    /* x[4] Internal state hysteresis*/
    dx[0] = x[2];
    dx[1] = x[3];
    dx[2] = (Kin[0]*u[0]+(tau_h*c_c/N[0])-tau_f)/JmKR[0];
    dx[3] = (-tau_h-be[0]*x[3])/Jje[0];
}

/* Output equations. */
void compute_y(double y[], double x[])
{
    y[0] = x[0];
    y[1] = x[1];
    y[2] = x[2];
    y[3] = x[3];
}
```

Annexes.

1.1.3.2 Identification script for HD model with cubic NL elasticity and trigonometric friction.

```
%% COMPILE THE FILE CONTAINING THE MODEL
% clc, clear;
addpath('C:\Users\Rafael C\Desktop\Ident_high_freq')
run ('script_3experiments');
mex modell.c
% mex modell_3out.c
%% INPUT, OUTPUT AND STATE NAME AND UNITS DEFINITIONS
InputName = {'DAC voltage'};
InputUnit = {'V'};
StateName = {'theta'; ... %Motor angle
             'q'; ... %Link angle
             'th_p'; ... %Motor speed
             'q_p'; ... %Link speed
             };
StateUnit = {'rad'; 'rad'; 'rad/s';...
            'rad/s';...
            };
OutputName = StateName(1:4);
OutputUnit = StateUnit(1:4);
%% PARAMETERS
n_p = 13; % Number of parameters
ParName = {'Motor inertia'; ...
           'Link side inertia '; ...
           'HarmonicDrive reduction ratio'; ...
           'Motor viscous friction';...
           'Spring Damping';...
           'Link viscous friction';...
           'Input gain';...
           'Spring stiffness constant';...
           'Coulomb friction';...
           'k3';...
           'striebeck';...
           'alpha';...
           'beta';...
           };
ParUnit = {'kgm^2'; 'kgm^2'; '' ; ...
          'Nms/rad'; 'Nms/rad'; 'Nms/rad'; 'Nm/A'; 'Nm/rad'; '' ;...
          'k3';...
          '' ; '' ; '' ;...
          };
Bm0 = 0.299278;
%PARAMETER INITIALIZATION AND BOUNDARIES
%           Jt     Jje     N     Bm     D     Be     Kin     K     tc     K3
ParValue={0.0073; 0.3; 70; Bm0; 0.05; 0.1; 14.4; 40 ;0.315 ; 150;...
          1.60667; 0.465822; 0.0445634;...
          };
ParMin={0.00036543; 0.01; 70; Bm0-0.05;0.0450; eps(0); 14.025; 37;0.28;
149;...
1.36; 0.04; 0.004;...
          };
ParMax={0.9;1; 70; Bm0+0.05; 0.075; 0.7; 14.975; 43;2.75;151;...
10; 1.5; 1.05;...
          };
```

Annexes.

```
% INITIAL STATES OF THE EXPERIMENT
%Initial state instant
val_state = 10;
ends = 12000;
vell = vell_f;%vell_f; vell_noise;
vell_b = vell_f_b;%vell_noise_b;
vel2 = vel2_f;%vel2_noise;
vel2_b = vel2_f_b;%vel2_noise_b;
vell_c = vell_f_c;
vel2_c = vel2_f_c;
%InitialStates Data set 1
InitialStates = {encoders1(val_state) ; -encoders2(val_state);
vell(val_state); -vel2(val_state)};
InitialStates = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates, ...
    'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);
%InitialStates Data set 2
InitialStates_b = {encoders1_b(val_state) ; -encoders2_b(val_state) ;
vell_b(val_state); -vel2_b(val_state)};
InitialStates_b = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates_b, ...
    'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);
% Create the idnlgrey object and the iddata object
FileName      = 'modell';           % File describing the model
structure.
Order         = [4 1 4];           % Model orders [ny nu nx].
Parameters    = struct('Name', ParName, 'Unit', ParUnit, 'Value',
ParValue, ...
    'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);
Ts            = 0;
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, Ts, ...
    'Name', 'Respuesta simulada', 'InputName', InputName, ...
    'InputUnit', InputUnit, 'OutputName', OutputName, ...
    'OutputUnit', OutputUnit, 'TimeUnit', 's');
%Adapt the data set sample time before putting it into the iddata object
z = iddata({[encoders1(val_state:ends) ' -encoders2(val_state:ends) '
vell(val_state:ends) -vel2(val_state:ends)],...
    [encoders1_b(val_state:ends) ' -encoders2_b(val_state:ends) '
vell_b(val_state:ends) -vel2_b(val_state:ends)]...
    [encoders1_c(val_state:ends) ' -encoders2_c(val_state:ends) '
vell_c(val_state:ends) -vel2_c(val_state:ends)]...
    }, {tension_s(val_state:ends)-2.5
tension_s_b(val_state:ends)-2.5...
    tension_s_c(val_state:ends)-2.5...
    }, 0.001);

z.Name = 'Datos del experimento';
z.InputName = nlgr.InputName;
z.InputUnit = nlgr.InputUnit;
z.OutputName = nlgr.OutputName;
z.OutputUnit = nlgr.OutputUnit;
z.ExperimentName = {'Estimation1' 'Estimation2'...
    'Validation'...
};
z.Tstart = 0;
z.TimeUnit = 's';
present(z)
```

Annexes.

```
%% Identification of the parameters with the model structure specified
in the idnlgrey object
%% First data set: Estimation
% Parameters(1).Fixed = true;
% Parameters(2).Fixed = true;
Parameters(3).Fixed = true;
Parameters(4).Fixed = true;
% Parameters(5).Fixed = true;
% Parameters(6).Fixed = true;
Parameters(7).Fixed = true;
% Parameters(8).Fixed = true;
% Parameters(9).Fixed = true;
% Parameters(10).Fixed = true;
% Estimation options (Searchmethod, max iterations...)
% Searchmethod = 'lm'; % 'lsqnonlin' 'gna' 'lm'
% nlgr.SimulationOptions.RelTol = 1e-5;
opt = nlgreyestOptions('SearchMethod', 'lm', 'Display', 'on');
opt.SearchOption.MaxIter = 15;
% First data set estimation
nlgr1 = nlgreyest(getexp(z,1), nlgr, opt);

%% Second data set: Validation 1
% Get estimated parameters in the first step
for i=1:n_p
ParValue(i)={nlgr1.Parameters(i).Value};
end

% Introduce the parameters (ParValue) in the new structure (Parameters)
for the next estimation
Parameters = struct('Name', ParName, 'Unit', ParUnit, 'Value',
ParValue, ...
'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);
% Fix parameters from nlgr
% Parameters(1).Fixed = true;
% Parameters(2).Fixed = true;
Parameters(3).Fixed = true;
Parameters(4).Fixed = true;
% Parameters(5).Fixed = true;
% Parameters(6).Fixed = true;
Parameters(7).Fixed = true;
% Parameters(8).Fixed = true;
% Parameters(9).Fixed = true;
% Parameters(10).Fixed = true;

% Build the idnlgrey object with the parameters estimated in the last
step and
% with the new initial states
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates_b, Ts,
...
'Name', 'Respuesta simulada', 'InputName', InputName, ...
'InputUnit', InputUnit, 'OutputName', OutputName, ...
'OutputUnit', OutputUnit, 'TimeUnit', 's');
```

Annexes.

```
% Estimate again with the second data set and the parameters obtained
in the previous step.
opt = nlgreyestOptions('SearchMethod', 'lm', 'Display', 'on');
opt.SearchOption.MaxIter = 15;
% opt1 = nlgreyestOptions('SearchMethod', 'gna', 'Display', 'on');
% opt1.SearchOption.MaxIter = 2;

%Same initial parameters with different search methods
nlgr2 = nlgreyest(getexp(z,1), nlgr, opt);
% nlgr1 = nlgreyest(getexp(z,2), nlgr, opt1);

%% Third data set : Validation
% Plot data and compare to identified NLGR and parameter result
evaluation
%% Compare the estimated model response with the data sets
n_ds = 3; % Number of data sets
set(gcf, 'DefaultLegendLocation', 'southeast');
figure
compare(getexp(z,n_ds), nlgr1, nlgr2);
figure
compare(getexp(z,n_ds-1), nlgr1, nlgr2);
%% 1 step prediction error
% figure
% pe(getexp(z,n_ds), nlgr, 'b');

figure
pe(getexp(z,n_ds), nlgr2, 'k');

% Check residuals
figure('Name', [nlgr.Name ': residuals of estimated model']);
resid(getexp(z,n_ds), nlgr2);

% figure
% resid(getexp(z,n_ds), nlgr1, 'r');

%% Check step response and covariance
figure
step(nlgr2);

getcov(nlgr2)
nlgr2.NoiseVariance

present(nlgr1);
present(nlgr2);
% present(nlgr3);
```


Annexes.

1.1.4 Harmonic drive model with nonlinear elasticity, trigonometric friction and hysteresis.

1.1.4.1 H.D model (Cubic elasticity, Bouc-Wen hysteresis, trigonometric friction)

```
void compute_dx(double *dx, double *x, double *u, double **p)
{
    /* Declaration of model parameters and intermediate variables. */
    double *JmKR, *Jje, *N, *D, *be, *bmKw, *Kin, *k1, *tcKR, *A, *B,
    *n, *w, *k3, *Fs, *alpha_f, *beta_f;

    double delta_e, delta_ep, tau_h, tau_f, k_nl;
    double c_c = 2.3/0.217; // Conversion from torque to voltage with
motor constants
    /* Retrieve model parameters. */
    JmKR = p[0]; /* Momento de inercia del motor
    Jje = p[1]; /* Momento de inercia del elemento
    N = p[2]; /* Factor de reduccion del HarmonicDrive
    bmKw=p[3];
    D=p[4];
    be=p[5];
    Kin=p[6];
    k1 = p[7];
    tcKR = p[8];
    /*Hysteresis parameters*/
    A = p[9];
    n = p[10];
    B = p[11];
    w = p[12];
    k3 = p[13];
    C = p[14];
    Fs = p[15];
    alpha_f = p[16];
    beta_f = p[17];
    /*Ecuaciones + NL elasticidad*/
    /*Torsional angle*/
    delta_e = x[1]-x[0]/N[0]; // Torsional angle
    delta_ep = x[3]-x[2]/N[0]; // Torsional speed
    k_nl = k1[0]*delta_e+k3[0]*pow(delta_e,3);
    tau_h = w[0]*k_nl+D[0]*delta_ep+(1-w[0])*k_nl*x[4];
    tau_f =
bmKw[0]*x[2]+(tcKR[0]+Fs[0]/cosh(alpha_f[0]*x[2])*tanh(beta_f[0]*x[2]));
    /* State equations. */
    /* x[0] angulo del motor */
    /* x[1] angulo del elemento*/
    /* x[2] velocidad del motor*/
    /* x[3] velocidad del elemento*/
    /* x[4] Internal state hysteresis*/
    dx[0] = x[2];
    dx[1] = x[3];
    dx[2] = (Kin[0]*u[0]+(tau_h*c_c/N[0])-tau_f)/JmKR[0];
    dx[3] = (-tau_h-be[0]*x[3])/Jje[0];
    dx[4] = (A[0]-pow(x[4],n[0])*B[0])*x[2];
}
```

Annexes.

```
/* Output equations. */  
void compute_y(double y[], double x[])  
{  
    y[0] = x[0];  
    y[1] = x[1];  
    y[2] = x[2];  
    y[3] = x[3];  
}
```

1.1.3.2 Identification script for HD model with cubicNL elasticity, hysteresis and trigonometric friction.

```
%% COMPILER THE FILE CONTAINING THE MODEL
clc, clear;
addpath('C:\Users\Rafael C\Desktop\Complete_model_ident')
run ('script_3experiments');
mex modell_knl_h.c

%% INPUT, OUTPUT AND STATE NAME AND UNITS DEFINITIONS
InputName = {'DAC voltage'};
InputUnit = {'V'};

StateName = {'theta'; ... %Motor angle
             'q'; ... %Link angle
             'th_p'; ... %Motor speed
             'q_p'; ... %Link speed
             'z'; ... %Hysteresis internal state
            };

StateUnit = {'rad'; 'rad'; 'rad/s';...
            'rad/s'; ''};...
            };

OutputName = StateName(1:4);
OutputUnit = StateUnit(1:4);

%% PARAMETERS
n_p = 17; % Number of parameters

ParName = {'Motor inertia'; ...
           'Link side inertia '; ...
           'HarmonicDrive reduction ratio'; ...
           'Motor viscous friction';...
           'Spring Damping';...
           'Link viscous friction';...
           'Current to torque motor constant';...
           'Spring stiffness constant 1';...
           'Coulomb friction';...
           'A';...
           'n';...
           'B';...
           'w';...
           'Spring stiffness constant 3';...
           'Fs';...
           'alpha';...
           'beta';...
          };

ParUnit = {'kgm^2'; 'kgm^2'; '' ; ...
          'Nms/rad'; 'Nms/rad'; 'Nms/rad'; 'Nm/A'; 'Nm/rad'; ''};...
          'Nm'; '''; ''';...
          '''; '''; '''; '''; ''';...
          };
```

Annexes.

```
Bm0 = 0.299278;

%PARAMETER INITIALIZATION AND BOUNDARIES
%      Jt      Jje      N      Bm      D      Be      Kin      Kl      tc
ParValue={0.00736543; 0.03; 70; Bm0; 0; 0.001; 14.4; 650 ;0.3007 ;...
          1.7218e-04; 0.1; 0.2; ...
          0.2; 2; 0.2; 0.5; 110;...
          };

ParMin={0.004; 0; 70;Bm0-0.1;0; eps(0); 14.325; 10;0.28;...
        1e-4; 0.0001; 0.0001 ; ...
        0.00001; 1; 0.001; 0; 0.001;...
        };

ParMax={0.009;10; 70; Bm0+0.1; 0; 0.009; 14.475; 80000;0.315;...
        2.5e-4; 100; 100; ...
        200; 3; 200; 1; 20000;...
        };

Parameters(3).Fixed = true;

% INITIAL STATES OF THE EXPERIMENT
%Initial state instant
val_state = 10;
vell = vell_f;%vell_f; vell_noise;
vell_b = vell_f_b;%vell_noise_b;
vel2 = vel2_f;%vel2_noise;
vel2_b = vel2_f_b;%vel2_noise_b;
vell_c = vell_f_c;
vel2_c = vel2_f_c;

%InitialStates Data set 1
InitialStates = {encoders1(val_state) ; -encoders2(val_state);
vell(val_state); -vel2(val_state);0};
InitialStates = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates, ...
'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);

%InitialStates Data set 2
InitialStates_b = {encoders1_b(val_state) ; -encoders2_b(val_state) ;
vell_b(val_state); -vel2_b(val_state);0};
InitialStates_b = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates_b, ...
'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);

% Create the idnlgrey object and the iddata object
FileName = 'modell_knl_h'; % File describing the
model structure.
Order = [4 1 5]; % Model orders [ny nu nx].
Parameters = struct('Name', ParName, 'Unit', ParUnit, 'Value',
ParValue, ...
'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);

Ts = 0;
```

Annexes.

```
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, Ts, ...
    'Name', 'Respuesta simulada', 'InputName', InputName, ...
    'InputUnit', InputUnit, 'OutputName', OutputName, ...
    'OutputUnit', OutputUnit, 'TimeUnit', 's');

%Adapt the data set sample time before putting it into the iddata object
z = iddata([encoders1(val_state:end)' -encoders2(val_state:end)'
    vel1(val_state:end) -vel2(val_state:end)],...
    [encoders1_b(val_state:end)' -encoders2_b(val_state:end)'
    vel1_b(val_state:end) -vel2_b(val_state:end)]...
    [encoders1_c(val_state:end)' -encoders2_c(val_state:end)'
    vel1_c(val_state:end) -vel2_c(val_state:end)]...
    ), {tension_s(val_state:end)-2.5 tension_s_b(val_state:end)-
    2.5...
    tension_s_c(val_state:end)-2.5...
    }, 0.001);

z.Name = 'Datos del experimento';
z.InputName = nlgr.InputName;
z.InputUnit = nlgr.InputUnit;
z.OutputName = nlgr.OutputName;
z.OutputUnit = nlgr.OutputUnit;
z.ExperimentName = {'Estimation' 'Validation1'...
    'Validation2'...
};
z.Tstart = 0;
z.TimeUnit = 's';
present(z)

% Identification of the parameters with the model structure specified
in the idnlgrey object
% First data set: Estimation 1
% Estimation options (Searchmethod, max iterations...)
Searchmethod = 'lm'; % 'lsqnonlin' 'gna' 'lm'
% nlgr.SimulationOptions.RelTol = 1e-5;
opt = nlgreyestOptions('SearchMethod', Searchmethod, 'Display', 'on');
opt.SearchOption.MaxIter = 15;
% First data set estimation
nlgr1 = nlgreyest(getexp(z,1), nlgr, opt);

% Second data set: Estimation 2
% Get estimated parameters in the first step
for i=1:n_p
    ParValue(i)={nlgr1.Parameters(i).Value};
end

% Introduce the parameters (ParValue) in the new structure (Parameters)
for the next estimation
Parameters = struct('Name', ParName, 'Unit', ParUnit, 'Value',
    ParValue, ...
    'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);
% Fix parameters from nlgr
Parameters(3).Fixed = true;
```

Annexes.

```
% Build the idnlgrey object with the parameters estimated in the last
step and
% with the new initial states
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates_b, Ts,
...
    'Name', 'Respuesta simulada', 'InputName', InputName, ...
    'InputUnit', InputUnit, 'OutputName', OutputName, ...
    'OutputUnit', OutputUnit, 'TimeUnit', 's');

% Estimate again with the second data set and the parameters obtained
in the previous step.
opt = nlgreyestOptions('SearchMethod', 'lsqnonlin', 'Display', 'on');
opt.SearchOption.MaxIter = 25;
% opt1 = nlgreyestOptions('SearchMethod', 'gna', 'Display', 'on');
% opt1.SearchOption.MaxIter = 2;

%Same initial parameters with different search methods
nlgr2 = nlgreyest(getexp(z,2), nlgr, opt);
% nlgr1 = nlgreyest(getexp(z,2), nlgr, opt1);

%% Third data set : Validation
%% Plot data and compare to identified NLGR and parameter result
evaluation
%% Compare the estimated model response with the 3 data sets
n_ds = 3; % Number of data sets
set(gcf, 'DefaultLegendLocation', 'southeast');
figure
compare(getexp(z,n_ds), nlgr1,nlgr2);

%% 1 step prediction error
% figure
% pe(getexp(z,n_ds), nlgr,'b');

% figure
% pe(getexp(z,n_ds), nlgr1,'k');

%% Check residuals
% figure('Name',[nlgr.Name ': residuals of estimated model']);
% resid(getexp(z,n_ds),nlgr);

% figure
% resid(getexp(z,n_ds),nlgr1,'r');

%% Check step response and covariance
% figure
% step(nlgr);
%
% getcov(nlgr)
% nlgr.NoiseVariance

present(nlgr1);
present(nlgr2);
```

Annexes.

1.2 Current control

1.2.3 Harmonic drive model with nonlinear elasticity and trigonometric friction.

1.2.3.1 H.D model (Cubic elasticity, no hysteresis, trigonometric friction)

```
void compute_dx(double *dx, double *x, double *u, double **p)
{
    /* Declaration of model parameters and intermediate variables. */
    double *Jm, *Je, *N, *D, *be, *bm, *Kin, *k1, *tc, *Fs, *alpha_f,
    *beta_f, *k3;
    double delta_e, delta_ep, tau_e, tau_f, k_nl;
    /* Retrieve model parameters. */
    Jm = p[0]; /* Momento de inercia del motor
    Je = p[1]; /* Momento de inercia del elemento
    N = p[2]; /* Factor de reduccion del HarmonicDrive
    bm=p[3];
    D=p[4];
    be=p[5];
    Kin=p[6];
    k1 = p[7];
    tc = p[8];
    Fs = p[9];
    alpha_f = p[10];
    beta_f = p[11];
    k3 = p[12];
    /*Ecuaciones + NL elasticidad*/
    /*Torsional angle*/
    delta_e = x[1]-x[0]/N[0]; // Torsional angle
    delta_ep = x[3]-x[2]/N[0]; // Torsional speed
    k_nl = k1[0]*delta_e+k3[0]*pow(delta_e,3);
    tau_e = k_nl+D[0]*delta_ep; // Linear elastic torque
    tau_f =
    bm[0]*x[2]+(tc[0]+Fs[0]/cosh(alpha_f[0]*x[2])*tanh(beta_f[0]*x[2]));
    /* State equations. */
    /* x[0] angulo del motor */
    /* x[1] angulo del elemento*/
    /* x[2] velocidad del motor*/
    /* x[3] velocidad del elemento*/
    dx[0] = x[2];
    dx[1] = x[3];
    dx[2] = (Kin[0]*u[0]+tau_e/N[0]-tau_f)/Jm[0];
    dx[3] = (-tau_e-be[0]*x[3])/Je[0];
}

/* Output equations. */
void compute_y(double y[], double x[])
{
    y[0] = x[0];
    y[1] = x[1];
    y[2] = x[2];
    y[3] = x[3];
}
```

Annexes.

1.2.3.2 Identification script for HD model with cubicNL elasticity, hysteresis and trigonometric friction.

```
%% COMPILE THE FILE CONTAINING THE MODEL
addpath('C:\Users\Rafael C\Desktop\SINGLESINUSCURRENT')
mex_model_current.c
run('script_3experiments.m')
%% INPUT, OUTPUT AND STATE DEFINITIONS
InputName = {'DAC'};
InputUnit = {'V'};
StateName = {'theta'; ... %Motor angle
             'q'; ... %Link angle
             'th_p'; ... %Motor speed
             'q_p'; ... %Link speed
             };
StateUnit = {'rad'; 'rad'; 'rad/s'; 'rad/s'};
OutputName = StateName(1:4);
OutputUnit = StateUnit(1:4);
%% PARAMETERS
n_p = 13;
ParName = {'Motor side inertia'; ...
           'Link side inertia'; ...
           'HarmonicDrive reduction ratio'; ...
           'Motor viscous friction';...
           'Spring Damping';...
           'Link viscous friction';...
           'Input constant';...
           'Spring stiffness constant';...
           'Coulomb friction';...
           'Striebeck';...
           'Alpha_f';...
           'Beta_f';...
           'k3';...
           };
ParUnit = {'kgm^2'; 'kgm^2'; '' ; ...
           'Nms/rad'; 'Nms/rad'; 'Nms/rad'; 'Nm/A'; 'Nm/rad';...
           'Nm/rad';...
           '' ; '' ; '' ; '' ;...
           };
Kin = 1.78/2.5*0.217*2; %P7
tc = 0.0297;%previous steps 0.03
Bm = 0.0064; %P4
%%PARAMETER INITIALIZATION AND BOUNDARIES
% Jm Je N Bm D Be Kin K1
ParValue={6.8874e-4; 0.0215; 70; Bm; 0.0562; 0.1538; Kin;
40.4364;...
tc; 0.398629; 0.2; 0.399435; 150;... % tc , Fs , alpha, beta
};
ParMin={0.00000306; 0; 70; 0; 1e-6; eps(0); 0.002; 2;...
0.0001; 0.0004; 0.0001; 0.00001; 140;...
};
ParMax={ 1; 10; 70; 100; 1000; 2; 1000;
90000;...
0.4; 200; 200; 200; 151;...
};
```


Annexes.

```
%% INITIAL STATES OF THE EXPERIMENT
%Initial state instant
val_state = 10;
vell = vell_f;%vell_f; vell_noise;
vell_b = vell_f_b;%vell_noise_b;
vel2 = vel2_f;%vel2_noise;
vel2_b = vel2_f_b;%vel2_noise_b;
% vell_c = vell_f_c;
% vel2_c = vel2_f_c;

%InitialStates Data set 1
InitialStates = {encoders1(val_state) ; -encoders2(val_state);
vell(val_state); -vel2(val_state)};
InitialStates = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates, ...
    'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);

%InitialStates Data set 2
InitialStates_b = {encoders1_b(val_state) ; -encoders2_b(val_state) ;
vell_b(val_state); -vel2_b(val_state)};
InitialStates_b = struct('Name', StateName, 'Unit', StateUnit, 'Value',
InitialStates_b, ...
    'Minimum', -Inf, 'Maximum', Inf, 'Fixed', true);

%% Create the idnlgrey object and the iddata object
FileName      = 'model_current';           % File describing the
model structure.
Order         = [4 1 4];                 % Model orders [ny nu nx].

Parameters    = struct('Name', ParName, 'Unit', ParUnit, 'Value',
ParValue, ...
    'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);

Parameters(3).Fixed = true;
Parameters(4).Fixed = true;
Parameters(5).Fixed = true;
Parameters(8).Fixed = true;
% Parameters(9).Fixed = true;
Parameters(13).Fixed = true;

Ts           = 0;

nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, Ts, ...
    'Name', 'Respuesta simulada', 'InputName', InputName, ...
    'InputUnit', InputUnit, 'OutputName', OutputName, ...
    'OutputUnit', OutputUnit, 'TimeUnit', 's');
```

Annexes.

```
%Adapt the data set sample time before putting it into the iddata object
z = iddata([encoders1(val_state:end)' -encoders2(val_state:end)'
vell1(val_state:end) -vel2(val_state:end)],...
          [encoders1_b(val_state:end)' -encoders2_b(val_state:end)'
vell1_b(val_state:end) -vel2_b(val_state:end)]...
          [encoders1(val_state:end)' -encoders2(val_state:end)'
vell1(val_state:end) -vel2(val_state:end)]...
          ),{tension_s(val_state:end)-2.5 tension_s_b(val_state:end)-
2.5...
          tension_s(val_state:end)-2.5...
          }, 0.001);

z.Name = 'Datos del experimento';
z.InputName = nlgr.InputName;
z.InputUnit = nlgr.InputUnit;
z.OutputName = nlgr.OutputName;
z.OutputUnit = nlgr.OutputUnit;
z.ExperimentName = {'Estimation1' 'Estimation2'...
'Validation'...
};
z.Tstart = 0;
z.TimeUnit = 's';
present(z)

% ESTIMATE THE MODEL
% nlgr.SimulationOptions.RelTol = 1e-5;

opt = nlgreyestOptions('SearchMethod', 'lm', 'Display', 'on');
opt.SearchOption.MaxIter = 20;

nlgr1 = nlgreyest(getexp(z,1), nlgr, opt);

% Second data set: Validation 1
% Get estimated parameters in the first step
for i=1:n_p
ParValue(i)={nlgr1.Parameters(i).Value};
end

% Introduce the parameters (ParValue) in the new structure (Parameters)
for the next estimation
Parameters = struct('Name', ParName, 'Unit', ParUnit, 'Value',
ParValue, ...
'Minimum', ParMin, 'Maximum', ParMax, 'Fixed', false);
% Fix parameters from nlgr
Parameters(3).Fixed = true;
Parameters(4).Fixed = true;
Parameters(5).Fixed = true;
Parameters(8).Fixed = true;
Parameters(9).Fixed = true;
Parameters(13).Fixed = true;
```

Annexes.

```
% Build the idnlgrey object with the parameters estimated in the last
step and
% with the new initial states
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates_b, Ts,
...
    'Name', 'Respuesta simulada', 'InputName', InputName, ...
    'InputUnit', InputUnit, 'OutputName', OutputName, ...
    'OutputUnit', OutputUnit, 'TimeUnit', 's');

% Estimate again with the second data set and the parameters obtained
in the previous step.
opt = nlgreyestOptions('SearchMethod', 'lm', 'Display', 'on');
opt.SearchOption.MaxIter = 20;
% opt1 = nlgreyestOptions('SearchMethod', 'gna', 'Display', 'on');
% opt1.SearchOption.MaxIter = 2;

%Same initial parameters with different search methods
nlgr2 = nlgreyest(getexp(z,2), nlgr, opt);
% nlgr1 = nlgreyest(getexp(z,2), nlgr, opt1);

%% Third data set : Validation
% Plot data and compare to identified NLGR and parameter result
evaluation
%% Compare the estimated model response with the data sets
n_ds = 3; % Number of data sets
set(gcf, 'DefaultLegendLocation', 'southeast');
figure
compare(getexp(z,n_ds-1), nlgr1, nlgr2);

figure
compare(getexp(z,n_ds), nlgr1, nlgr2);
%
% % 1 step prediction error
% figure
% pe(z, nlgr);
%
% % Check residuals
% figure('Name', [nlgr.Name ': residuals of estimated model']);
% resid(z,nlgr);
%
% % Check step response
% figure
% step(nlgr);
%
% % Check covariance
%
getcov(nlgr)
nlgr.NoiseVariance
present(nlgr1)
present(nlgr2)
```

2. Keil uVison code

2.1 Main.c

```
/* Include core modules */
#include "stm32f4xx.h"
/* Include my libraries here */
#include "defines.h"
#include "tm_stm32f4_delay.h"
#include "tm_stm32f4_disco.h"
#include "tm_stm32f4_usb_vcp.h"
#include "tm_stm32f4_sdram.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "encoder.h"
#include "init.h"
#include "led.h"
#include "fonctions.h"
extern volatile int8_t flag_Exp;
// MAIN PROGRAM
int main(void) {
int32_t Encoder1, Encoder2 =0;
int32_t Encoder1_anterior, Encoder2_anterior=0;
float Encoder1_rad, Encoder2_rad = 0;
float Encoder1_rad_previous, Encoder2_rad_previous = 0;
float vel_Encoder1, vel_Encoder2 =0;
float vel_Encoder1_anterior, vel_Encoder2_anterior = 0;
float Read, Read2, Read3, Read4, Read5, Read6;
char cadena;
```

Annexes.

```
char cadena;
uint32_t t, aux = 0;
float tension_value;
float vumbral = 1000;
float saturador, saturador2 =0;
float enc_offset = 205.22;
float enc_offset2 = 205.22+0.67;
float v_aux1, v_aux2;
float tiempo;
float tsampling = 0.001; // change for different controltimer frequency
int time_experiment = 8000 ; // change for different controltimer frequency
int time_s = time_experiment/1000; // change for different controltimer frequency
int tsignal;
//-----//
FULL_INIT_stm(); //System initialization
init_experiment(); // Experiment initialization
//-----//
//-----experiment start -----//
int i = 0;
for(i=0;i<=time_experiment;i++){
flag_Exp = 0;
tiempo = t*tsampling;
tsignal = tiempo;
Encoder1 = TIM_GetCounter(TIM8);
Encoder2 = TIM_GetCounter(TIM4);
Encoder1_rad = Encoder1*2*3.1415/2000; // *2 pi/resolution
Encoder2_rad = Encoder2*2*3.1415/2000; // *2 pi/resolution
//-----//
// if timer saturates, the speed equals the previous speed .
```

Annexes.

```
//-----//
// if timer saturates, the speed equals the previous speed .
if (Encoder1_rad - Encoder1_rad_previous >= 200 | Encoder1_rad_previous - Encoder1_rad >=200) {
// encoder 1
vel_Encoder1 = vel_Encoder1_anterior;}
else vel_Encoder1 = (Encoder1_rad - Encoder1_rad_previous) / tsampling;    // (rad/s)
if (Encoder2_rad - Encoder2_rad_previous >= 200 | Encoder2_rad_previous - Encoder2_rad >=200){
// encoder 2
vel_Encoder2 = vel_Encoder2_anterior;
}
else vel_Encoder2 = (Encoder2_rad - Encoder2_rad_previous) / tsampling;    // (rad/s)
//-----//
/*
// Corregir posición y volver a calcular la velocidad con posicion correcta

v_aux1 = (Encoder1_rad - Encoder1_rad_previous) / tsampling;
v_aux2 = (Encoder2_rad - Encoder2_rad_previous) / tsampling;
if(v_aux1 <- vumbral){
saturador += enc_offset;
}else if(v_aux1 > vumbral){
saturador -= enc_offset;
}
if(v_aux2 <- vumbral){
saturador2 += enc_offset2;
}else if(v_aux2 > vumbral){
saturador2 -= enc_offset2;    }
*/
//-----//
//-----+-- tension_value -----//
```

Annexes.

```
//----- tension_value -----//  
// Different signals as input for the experiments  
// Random number generation  
int r = rand() % 5;  
float urand = (r-2.5)*0.25;  
// Multisinus input  
float u = (sin(2*3.1415*tiempo*5) + 5*sin(2*3.1415*tiempo*15)+ sin(2*3.1415*tiempo*0.5) +  
0.75*cos(2*3.1415*2*tiempo) )/10 + 3.75;  
// Single sinus  
float sins = (1.5*sin(2*3.1415*tiempo))+2.5+urand;  
// Chirp signal  
float chirp = 0.325*cos(2*3.1415*0.1*tiempo+(50-0.1)*tiempo*tiempo/(2*4))+2.5;  
// Determining the signal sent  
if (tiempo<0.25)  
tension_value =2.5;  
if (tiempo>0.25)  
tension_value = u;  
Motor1_Voltage(tension_value); //sending the value to the motor  
//-----//  
//-----Writing the data into the SDRAM, ENCODER WRAP FIXED-----//  
TM_SDRAM_WriteFloat(aux, Encoder1_rad +saturador );  
TM_SDRAM_WriteFloat(aux+8, Encoder2_rad + saturador2 );  
TM_SDRAM_WriteFloat(aux+16, vel_Encoder1);  
TM_SDRAM_WriteFloat(aux+24, vel_Encoder2);  
TM_SDRAM_WriteFloat(aux+32, tension_value);  
TM_SDRAM_WriteFloat(aux+40, tiempo );  
aux += 48;  
  
vel_Encoder1_anterior = vel_Encoder1;
```

Annexes.

```
vel_Encoder1_anterior = vel_Encoder1;
Encoder1_anterior = Encoder1;
Encoder1_rad_previous = Encoder1_rad;

vel_Encoder2_anterior = vel_Encoder2;
Encoder2_anterior = Encoder2;
Encoder2_rad_previous = Encoder2_rad;

while(flag_Exp != 1);
t++;
}
end_experiment();

//-----//
//-----READING THE DATA FROM THE SDRAM AND SENDING IT BY USB -----//
aux = 0;
if (TM_USB_VCP_GetStatus() == TM_USB_VCP_CONNECTED) {
TM_USB_VCP_Puts("A = [");
for (i = 0; i < time_experiment; i++) {
Read = TM_SDRAM_ReadFloat(aux);
Read2 = TM_SDRAM_ReadFloat(aux+8);
Read3 = TM_SDRAM_ReadFloat(aux+16);
Read4 = TM_SDRAM_ReadFloat(aux+24);
Read5 = TM_SDRAM_ReadFloat(aux+32);
Read6 = TM_SDRAM_ReadFloat(aux+40);
aux += 48;

sprintf(&cadena,"%f %f %f %f %f %f \r\n", Read , Read2, Read3, Read4, Read5, Read6);
TM_USB_VCP_Puts(&cadena);
```


Annexes.

```
sprintf(&cadena,"%f %f %f %f %f %f \r\n", Read , Read2, Read3, Read4, Read5, Read6);
TM_USB_VCP_Puts(&cadena);
Delayms(5);
}
TM_USB_VCP_Puts("];");
}
//-----//
//-----//
while(1);
}
//_____
_____//
```

2.2 Fonctions.c

```
#include "stm32f4xx.h"          // Device header
// -----
void Motor1_Voltage(float value){
//   DAC->DHR12R1 = value*4095/3*3/5;
//   DAC->DHR12R2 = value*4095/3*3/5;
}
// -----
// -----
void enable_escon()              // Activation of PG15
// ESCON ON
    GPIO_ToggleBits(GPIOD, GPIO_Pin_15);
    GPIOD->ODR &= 0xFFFF2FFF;
    GPIOD->ODR |= 0x00002000;
    GPIOD->ODR &= 0xFFFFBFFF;
    GPIOD->ODR |= 0x00000000;
}
void disable_escon()            // desactivation of PG15
// ESCON OFF
    GPIOD->ODR &= 0xFFFF7FFF;
    GPIOD->ODR |= 0x00000000;
    GPIOD->ODR &= 0xFFFFBFFF;
    GPIOD->ODR |= 0x00004000;
    GPIOD->ODR &= 0xFFFFDFFF;
    GPIOD->ODR |= 0x00000000;
}
```

2.3 Init.c

```
/* Include core modules */
#include "stm32f4xx.h"
/* Include my libraries here */
#include "defines.h"
#include "tm_stm32f4_delay.h"
#include "tm_stm32f4_disco.h"
#include "tm_stm32f4_usb_vcp.h"
#include "tm_stm32f4_sdram.h"
#include "encoder.h"
#include "led.h"
#include "fonctions.h"
int on=1, off=0;
/*-----*/
void Control_TimerInit(uint32_t frequency){
    uint16_t PrescalerValue;
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    uint32_t ARR;
    uint32_t period;
    ARR = 84000000.0F / ((420+1)*frequency) -1;
    period = (int) ARR;
    if(ARR != period){
        period++;
    }// +1 en caso de numero fraccionario
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
    //Set the prescaler value
    PrescalerValue =420;
```

Annexes.

```
// Time base configuration
TIM_TimeBaseStructure.TIM_Period = period;
TIM_TimeBaseStructure.TIM_Prescaler = PrescalerValue;
TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

TIM_Cmd(TIM2, ENABLE);

// Enable the TIM2 global Interrupt
NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;

NVIC_Init(&NVIC_InitStructure);

}
/*-----*/
```

Annexes.

```
void dac_init(void){
    GPIO_InitTypeDef GPIO_InitStructure;
    DAC_InitTypeDef DAC_InitStructure;
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    /* GPIOA Periph clock enable */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
    /* GPIOD Periph clock enable */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
    /* Enable DAC and GPIOC clock */
    //RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC | RCC_APB1Periph_TIM2, ENABLE);
    /* Configure PA.04/05 (DAC) as output -----*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    /* DAC channel1 & DAC_Channel_2 Configuration */
    DAC_InitStructure.DAC_Trigger = DAC_Trigger_None;
    DAC_InitStructure.DAC_WaveGeneration = DAC_WaveGeneration_None;
    DAC_InitStructure.DAC_OutputBuffer = DAC_OutputBuffer_Enable;
    DAC_Init(DAC_Channel_2, &DAC_InitStructure);
    DAC_Init(DAC_Channel_1, &DAC_InitStructure);
    /* Enable DAC Channel1 & Channel2: Once the DAC is enabled, PA.05 is
    automatically connected to the DAC converter. */
    DAC_Cmd(DAC_Channel_2, ENABLE);
    DAC_Cmd(DAC_Channel_1, ENABLE);
}
/*-----*/
void LED_init(){
```

Annexes.

```
void LED_init(){
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOG, ENABLE);
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13 | GPIO_Pin_14;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_Init(GPIOG, &GPIO_InitStructure);
}

void bouton_init(){
    /* Enable clock for GPIOD */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
    /* Enable clock for SYSCFG */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
    /* Set pin as input */
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
    GPIO_InitStructure.GPIO_Speed = GPIO_Fast_Speed;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    /* Tell system that you will use PA0 for EXTI_Line0 */
    SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource0);
    /* PA0 is connected to EXTI_Line0 */
    EXTI_InitTypeDef EXTI_InitStructure;
    EXTI_InitStructure.EXTI_Line = EXTI_Line0;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE; /* Enable interrupt */
}
```

Annexes.

```
EXTI_InitStruct.EXTI_LineCmd = ENABLE; /* Enable interrupt */
EXTI_InitStruct.EXTI_Mode = EXTI_Mode_Interrupt; /* Interrupt mode */
EXTI_InitStruct.EXTI_Trigger = EXTI_Trigger_Rising; /* Triggers on rising and falling edge */
EXTI_Init(&EXTI_InitStruct); /* Add to EXTI */

// Enable the EXTI0 global Interrupt
NVIC_InitTypeDef NVIC_InitStructure;
NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x00;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

}

/* Set interrupt handlers */
/* Handle PA0 interrupt */
void EXTI0_IRQHandler() {
    /* Make sure that interrupt flag is set */
    if (EXTI_GetITStatus(EXTI_Line0) != RESET) {
        /* Do your stuff when PA0 is changed */
        GPIO_ToggleBits(GPIOG, GPIO_Pin_14 | GPIO_Pin_13);
        /* Clear interrupt flag */
        EXTI_ClearITPendingBit(EXTI_Line0);
    }
}

void homming_init(){
```

Annexes.

```
void FULL_INIT_stm(){
    SystemInit();
    encoder1_config();
    encoder2_config();
    encoder4_config();
    dac_init();
    TM_DELAY_Init();
    TM_USB_VCP_Init();
    TM_DISCO_LedInit();
    TM_SDRAM_Init();
    LED_init();
}
/*-----*/
void init_experiment(){
    desable_escon();
    Motor1_Voltage(2.5);          //2.5 for both directions (ESCON CONFIG OFFSET)
    // homming_init();
    Delays(10000);                // Time to open teraterm
    enable_escon();                // enable escon
    Control_TimerInit(1000); // Control de 1 ms
    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE); // timer config
}
/*-----*/
void end_experiment(){
    Motor1_Voltage(2.5); // le moteur ne bouge pas
    desable_escon();    // On désactive le Escon
    TIM_ITConfig(TIM2, TIM_IT_Update, DISABLE);
}
/*-----*/
```


2.4 Encoder.c

```
/* Include core modules */
#include "stm32f4xx.h"
/* Include my libraries here */
#include "fonctions.h"
void encoder1_config(void){
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM8, ENABLE);
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    //Timer AF Pins Configuration
    GPIO_PinAFConfig(GPIOC, GPIO_PinSource6, GPIO_AF_TIM8);
    GPIO_PinAFConfig(GPIOC, GPIO_PinSource7, GPIO_AF_TIM8);
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_TimeBaseStructure.TIM_Period = 0xffff;
    TIM_TimeBaseStructure.TIM_Prescaler = 0;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM8, &TIM_TimeBaseStructure);
    TIM_EncoderInterfaceConfig (TIM8, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising,
TIM_ICPolarity_Rising);
    TIM_SetAutoreload (TIM8, 0xffff);
    TIM_Cmd (TIM8, ENABLE);
}
```

Annexes.

```
// PD12 y PD13
void encoder2_config(void){
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
    //Timer AF Pins Configuration
    GPIO_PinAFConfig(GPIOD, GPIO_PinSource12, GPIO_AF_TIM4);
    GPIO_PinAFConfig(GPIOD, GPIO_PinSource13, GPIO_AF_TIM4);
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_TimeBaseStructure.TIM_Period = 0xffff;
    TIM_TimeBaseStructure.TIM_Prescaler = 0;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
    TIM_EncoderInterfaceConfig (TIM4, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising,
TIM_ICPolarity_Rising);
    TIM_SetAutoreload (TIM4, 0xffff);
    TIM_Cmd (TIM4, ENABLE);
}

// -----
// -----
```

Annexes.

```
void encoder4_config(){
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1, ENABLE);
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9 | GPIO_Pin_11;
    GPIO_Init(GPIOE, &GPIO_InitStructure);
    //Timer AF Pins Configuration
    GPIO_PinAFConfig(GPIOE, GPIO_PinSource9, GPIO_AF_TIM5);
    GPIO_PinAFConfig(GPIOE, GPIO_PinSource11, GPIO_AF_TIM5);
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_TimeBaseStructure.TIM_Period = 0xffff;
    TIM_TimeBaseStructure.TIM_Prescaler = 0;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM1, &TIM_TimeBaseStructure);
    TIM_EncoderInterfaceConfig (TIM1, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising,
TIM_ICPolarity_Rising);
    TIM_SetAutoreload (TIM1, 0xffff);
    TIM_Cmd (TIM1, ENABLE);
    /* Enable the TIM5 global Interrupt */
    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_InitStructure.NVIC_IRQChannel = TIM5_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

2.5 Stm32f4xx_it.c

```
#include "stm32f4xx_it.h"
#include "stm32f4xx_dac.h"
#include "main.h"
int8_t flag_Exp;
void TIM2_IRQHandler(void)
{
    if (TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
    {
        //flag_Control = 1;
        flag_Exp = 1;

        //Clear interruption flag
        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
    }
}
```

Annexes.