POLITECNICO DI TORINO

MECHANICAL AND AEROSPACE ENGINEERING DEPARTMENT

Master's Degree Thesis in Aerospace Engineering

Optimization Methodologies Study for the development of Prognostic Artificial Neural Networks

Candidate: GIUSEPPE PETTI

Supervisors: Prof. PAOLO MAGGIORE

Eng. MATTEO D. L. DALLA VEDOVA

Eng. GAETANO QUATTROCCHI

DECEMBER 2020

ACKNOWLEDGMENTS

ABSTRACT

In this work, I discuss the implementation and optimization of an Artificial Neural Network based on the analysis of the back-EMF coefficient capable of making ElectroMechanical Actuator (EMA) prognostics.

Aircraft manufacturers are increasingly focusing on the use of electromechanical actuators as they have numerous advantages in terms of weight and compactness respect to the technologies adopted at this time. However, they are early technology and for this reason, engineers are still studying the failure modes that characterize this component. The objective of this thesis is to study a methodology for the recognition of faults within the system.

To solve the problem my supervisors have thought of implementing a logic, based on artificial intelligence, particularly on artificial neural networks, which allows to estimate the remaining useful life of the system components starting from a training dataset. The neural network learns autonomously the relationships that link the quantities given as input with those in output.

However, during learning, the creators need to set the value of the hyperparameters. My job is to show how these values influence learning and how it is possible to optimize the network to make it more performing in terms of computational cost and complexity, so that the variation of hyperparameters improves supervised learning. The future of aviation is certainly based on the "more electric" philosophy. Electricity is the only indispensable energy source for an aircraft. Nowadays, the remote hypothesis of "full electric aircraft" is still under study and yet there are several queries to be clarified.

The results are very satisfactory considering the small number of examples present in the available dataset. In the future, I think that we can build a neural network having datasets with a greater number of examples and deeper even though this, as you can read in this/my thesis, does not always turn out to be an advantage. For this reason, optimizing the work is important.

Table of Contents

1. Introduction

1.1. Prognostics: Why and What it is

2. Electromechanical Actuator (EMA)

A servo actuator is a device used to control the position or velocity of a mechanical element by converting power from different sources (hydraulic, electrical, or pneumatic) into a controlled motion. [1]

An electromechanical actuator (EMA) converts electrical energy into mechanical energy necessary for the movement of the aircraft's control surfaces.

Generally, for the primary controls, hydraulic powering is used with linear cylinder-piston or, very rarely, rotary motors; the secondary controls can be realized by means of hydraulic or electric rotary motors. However, EMAs are now being studied to be implemented for primary flight controls.

An EMA consists of:

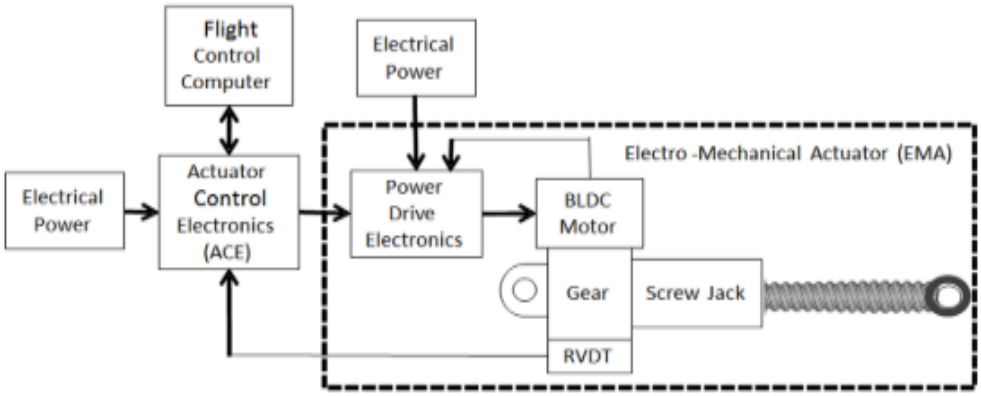


Figure 1: ElectroMechanical Actuator scheme

- ACE (Advanced Control Electronics): this control logic allows to calculate and output the error deriving from the difference of two signals, i.e. the FBW signal given by the pilot as a command and the position feedback signal read by the sensors.

- PDE (Power Drive Electronics): this module is powered by DC current and transforms the error deriving from the ACE into a modulated 3-phase AC current that will drive the electric motor.
- BLDC Motor: BrushLess Direct Current motor, converts electrical power into mechanical power.
- Gear: mechanical transmission that transforms low torque and high angular speed into high torque and low angular speed to move the actuator.
- Screw Jack: a device used to convert rotational motion into linear motion.
- RVDT: Rotary Variable Differential Transducer is an angular position acquisition sensor that allows the closure of the feedback loop.

The most important parameter of this device is the gear ratio τ:

$$\boldsymbol{\tau = \frac{\dot{\theta}_m}{\dot{\theta}_u} = \frac{T_m}{\eta\, T_u}}$$

Equation 2.1: Gear ratio

where $\dot{\theta}_m$ and T_m are respectively the velocity and the torque of the motor, $\dot{\theta}_u$ and T_u are the velocity and the torque of the user; finally, η is the transmission efficiency.

2.1. Model Description

A model on MATLAB-Simulink was used to test the prognostic algorithms and see the real-life response. The latter has several advantages. In this way it, is possible to reduce costs, develop the algorithm without having the physical system at hand and speed up its development. It is possible to simulate multiple malfunctions such as partial coil circuits.

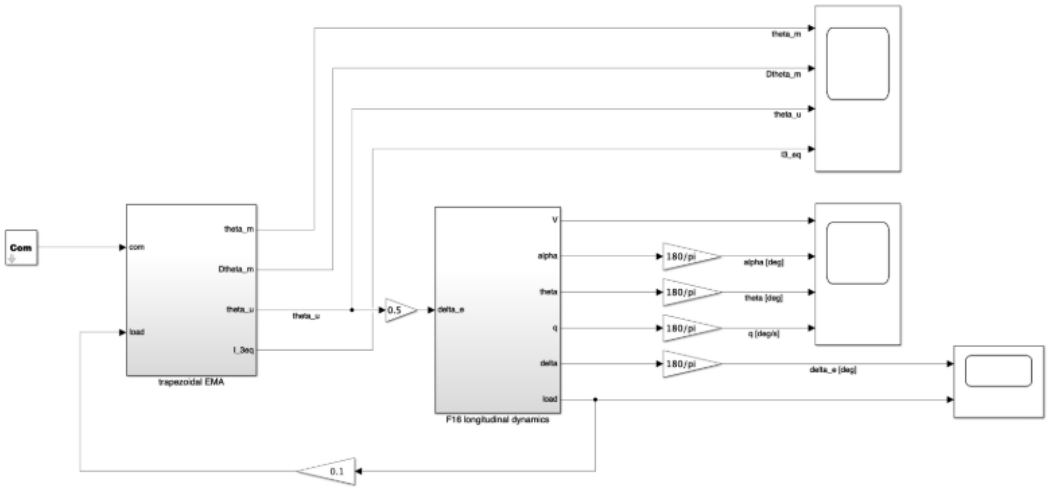


Figure 2: Simulink Model

On the left side we can see the *Com* block, which is the subsystem used to model the commanded position to be achieved by the system.

In the *trapezoidal EMA* block the whole electromechanical servomechanism is modeled, it receives the COM signal and the loads acting on the aircraft flight control surface as inputs and it gives as output δ_e, which is the deflection of the elevetor (assuming that the servomechanism is mounted on the aircraft's elevator).

Finally, in the *F16 longitudinal dynamics* block is modeled the dynamic response of an F16 aircraft passing through a model described in the state space form.

We now proceed with the accurate description of each block. [2]

2.2. Com subsystem

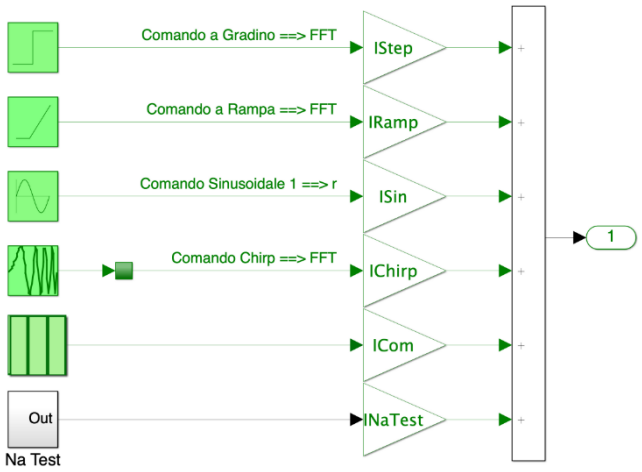


Figure 3: Com block scheme

This block allows the generation of different signals and simulates the FBW command given by the pilot that the control system must follow.

We have several commands available: a step command, a ramp command, a sinusoidal command, a chirp command that is sinusoidal with variable frequency and a command defined by the user. We can select the function we prefer by modifying some parameters.

2.3. Trapezoidal EMA subsystem

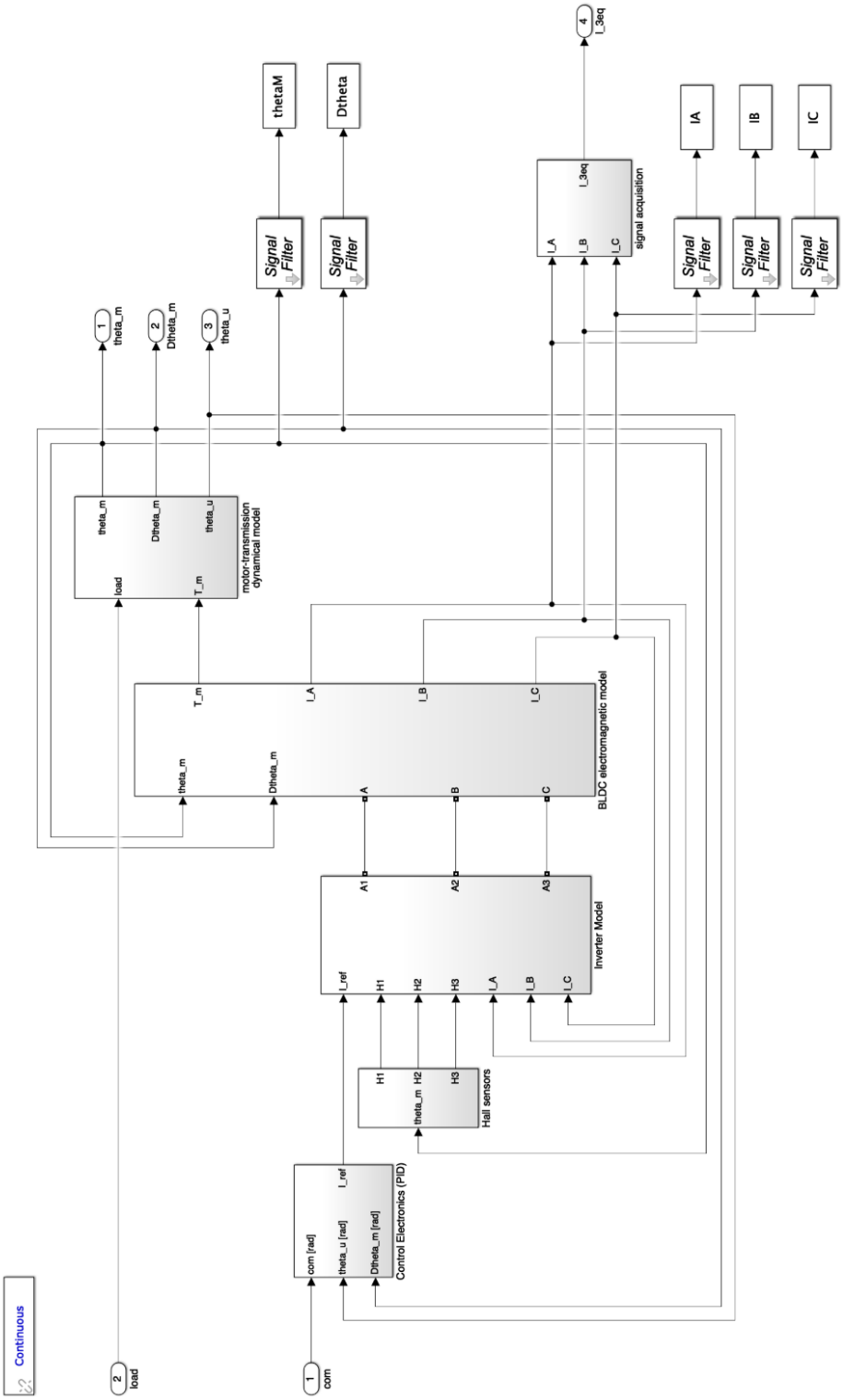


Figure 4: Trapezoidal EMA subsystem

In this subsystem, there are other subsystems which will now be analyzed.

The command generated by the COM block enters the PID subsystem, from which a reference current flow. The latter enters an inverter model in which the three-phase current that powers the BLDC motor is generated.

The motor torque comes out of the BLDC model, which enters the dynamic model of the transmission and the currents, that are read by an acquisition system, returns in feedback in the inverter model.

The dynamic model of the transmission outputs the position of the user, the speed and angular position of the engine. The first two are used in the PID controller for the determination of the reference current while the last two are used for the accomplishment of the BLDC electromagnetic motor's model. Finally, the position of the motor is read by the hall sensors. A low pass filter is used to read signals.

2.3.1. Control Electronics (PID) subsystem

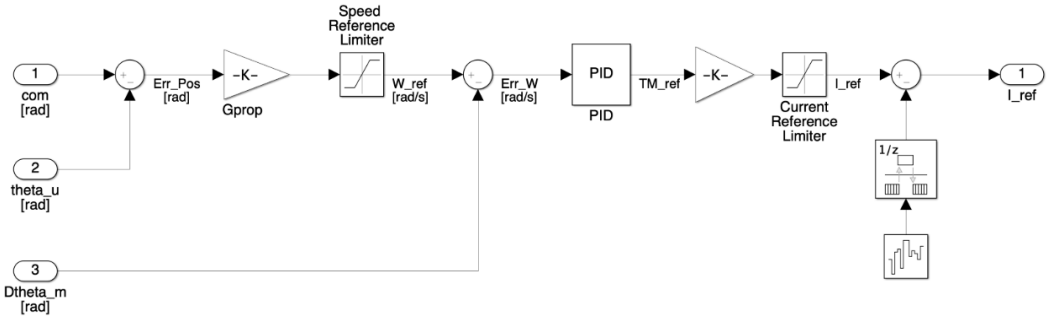


Figure 5: Control Electronics subsystem

Control Electronics subsystem is used to create the reference current that goes into the inverter model.

Starting from the command and the feedback signal of the user's position and speed, an error signal is generated which is given as input to a PID-type controller. A Proportional-Integral-Derivative controller is a control loop mechanism employing feedback signal. The controller attempts to minimize the error over time by adjustment of a control variable $u(t)$.

- P – The proportional term acts on the rise time and it reduces slightly the steady state error, however a high value leads to instability.
- I - The integrative term erases the steady state error but worsening the transistory response.
- D – The derivative term decreases the overshoot improving the transitory term. [3]

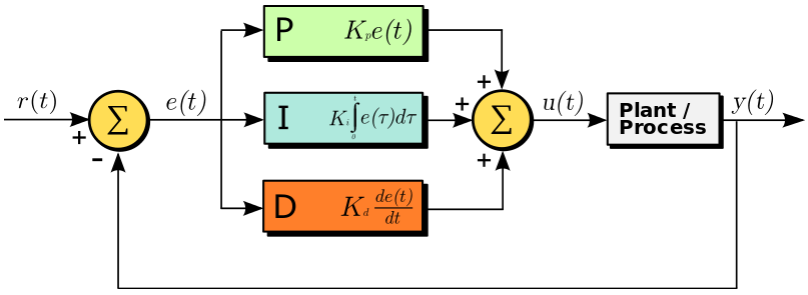


Figure 6: PID scheme

To have an optimal response it is necessary to fine-tune the coefficients K_P, K_I, K_D. Here are some effects to consider when choosing coefficients.

	Rise Time	Overshoot	Settling Time	Steady State Error	Stability
Increasing K_P	decrease	increase	small increase	decrease	degrade
Increasing K_I	small decrease	increase	increase	large decrease	degrade
Increasing K_D	small decrease	decrease	decrease	minor change	improve

Figure 7: PID tuning

In the model there are two saturations to safeguard the integrity of the motor's assembly, or rather to limit the currents and angular speeds. Background noise is added to recreate a realistic effect to the final signal.

2.3.2. Hall Sensors subsystem

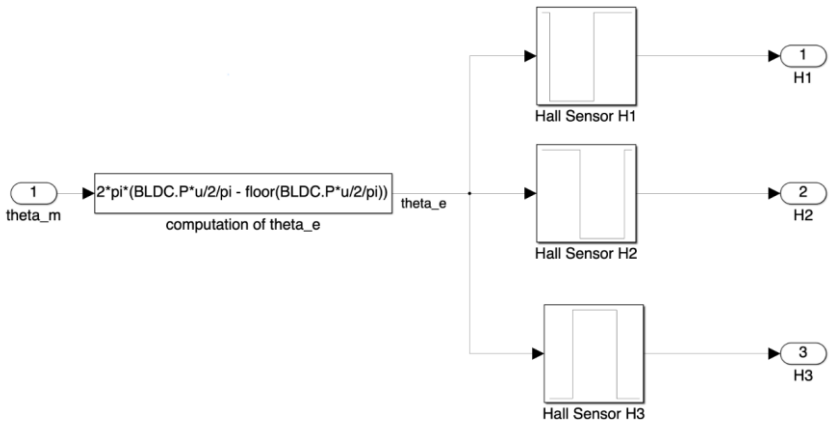


Figure 8: Hall Sensors subsystem

The block receives the angular position of the motor θ_m as input and returns three signals: $H1, H2\ and\ H3$.

These last ones can take the value of 0 or 1 according to a table of the mounted sensor in which the electrical angle is evaluated through the following relationship:

$$\boldsymbol{\theta_e = 2\pi\left[\frac{p\theta_m}{2\pi} - floor\left(\frac{p\theta_m}{2\pi}\right)\right]}$$

Equation 2.2: Relationship between electrical angle θ_e, motor's angular position θ_m and number of motor pole pairs p

2.3.3. Inverter Model subsystem

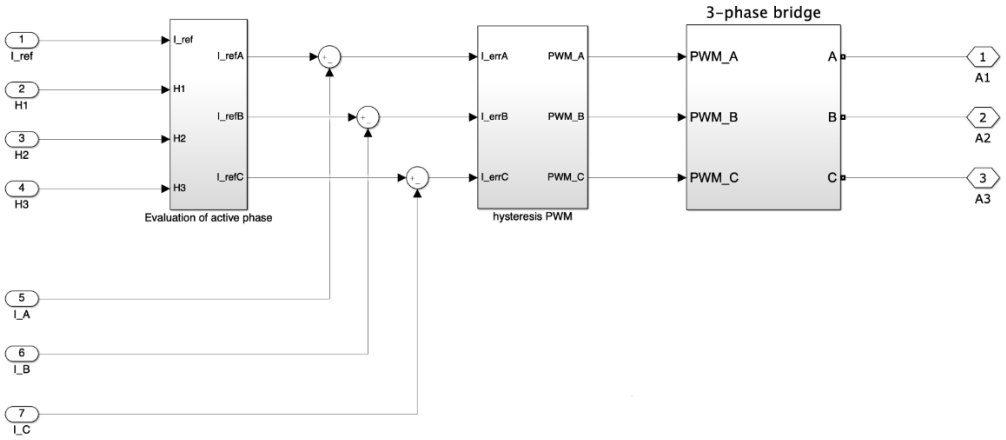


Figure 9: Inverter Model subsystem

This block is the most important of the whole model. It receives in input:

- the current I_ref, that is a pure control signal which has the task of controller to allow feedback control.

- H1, H2 and H3 or the signals that come directly from the modeling of the Hall sensors.

- I_A, I_B and I_C which are the values of the currents of each phase.

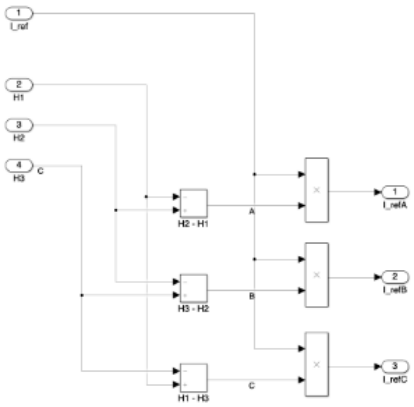


Figure 10: Evaluation of active system scheme

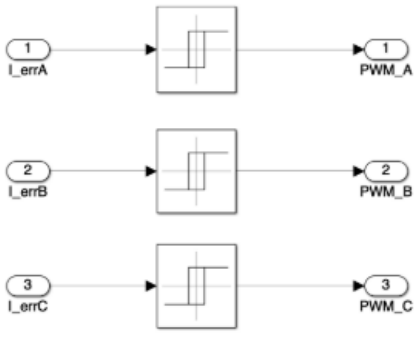


Figure 11: Hysteresis PWM scheme

Three reference currents useful for the switching logic are determined within the Evolution of active phase subsystem. They are calculated by adding and subtracting

the signals from the Hall sensors and multiplying these by the reference current. While in the Hysteresis PWM subsystem each reference current passes through a hysteresis block which returns the same value in inputs if the latter is not higher than $+hb$ or lower than $-hb$. If these hypotheses are satisfied it returns 1 or 0 respectively.

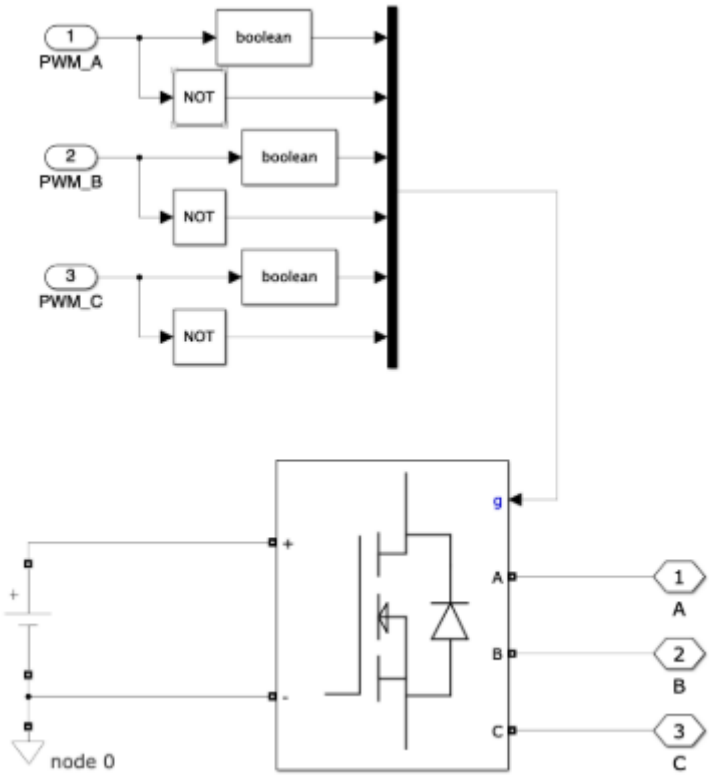


Figure 12: H-Bridge subsystem

Modeling of the H-bridge is done through the Simulink library through the "Universal bridge" block. To work, the block must be powered by a DC power and the PWM signals must enter inside according to a precise logic. For controlling the rotation direction of the motor, the direction of the current can be inverted and the most common method of doing that is by using an H-Bridge.

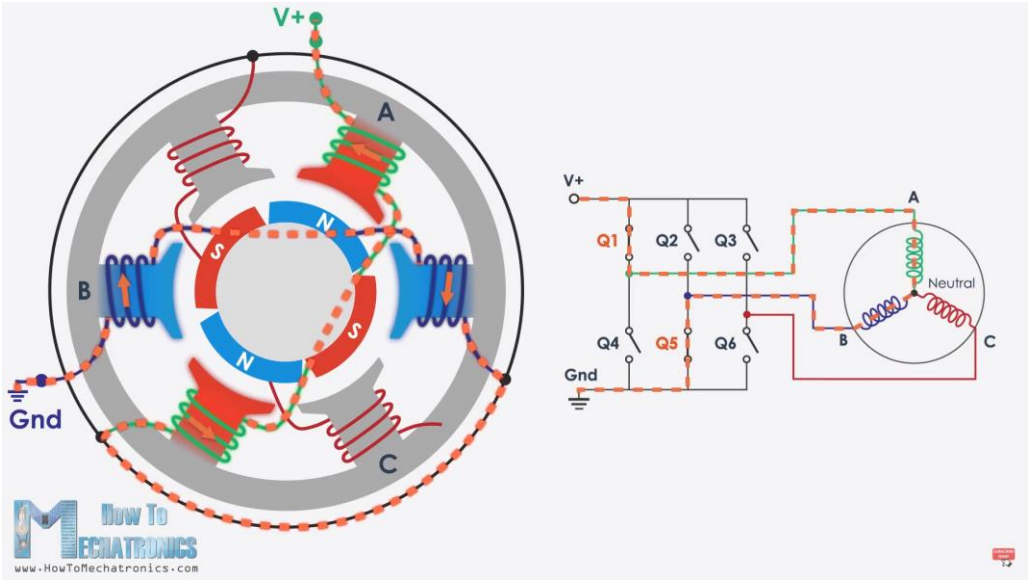


Figure 13:Combination of three separate H-Bridges for BLDC motors [4]

If we connect the phase A to the positive DC voltage, with some kind of switch like a MOSFET, and on the other side, connect the phase B to ground, then the current will flow from V+, through phase A, the neutral point and phase B, to ground. So, we generated the four different magnetic poles which cause the rotor to move. With this configuration we have a star connection of the motor phases, where the neutral point is internally connected. To allow a rotor rotation of 360 degrees it is necessary to activate the two correct MOSFETs in each of the 6 intervals. [4]

If we combine the methods of PWM with H-Bridge a BLDC motor can be fully controlled.

The output of the subsystem are the voltages $A, B\ and\ C$.

2.3.4. BLDC electromagnetic subsystem

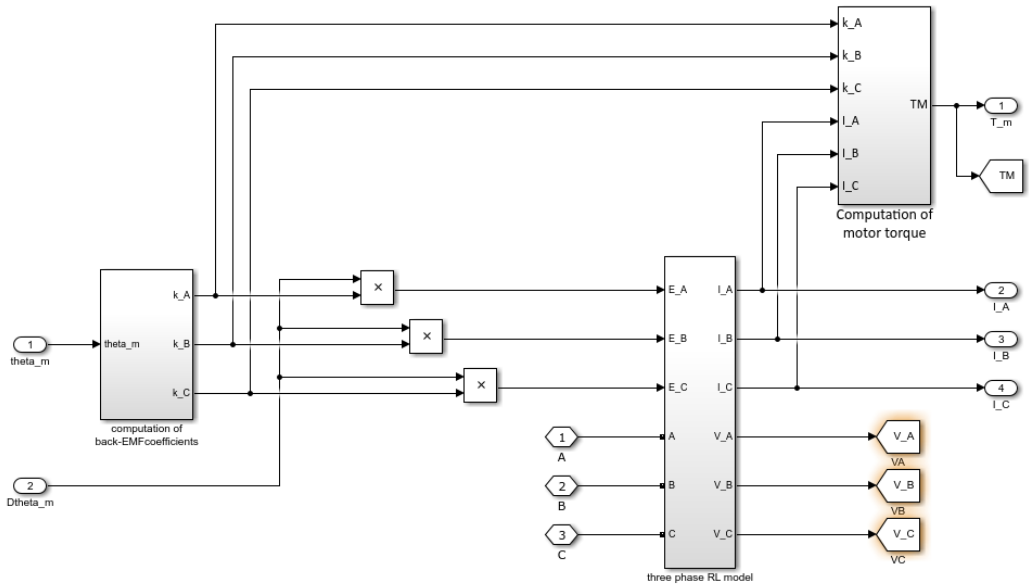


Figure 14: BLDC electromagnetic subsystem overview

This subsystem consists of three main blocks:

1. Computation of back-EMF coefficients block: it calculates the back-EMF coefficient as a function of angular position of the motor.

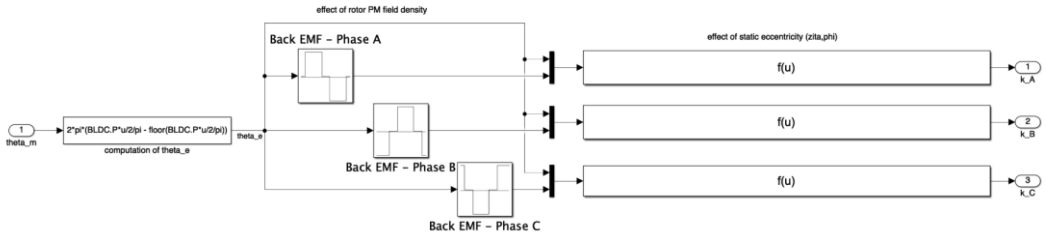


Figure 15: Computation of back-EMF coefficient block

$$k_{bemf}^{i} = k_{e}^{i}(\theta_m)\left(1 + \zeta\cos\left(\theta_m + \frac{2(i-1)}{3}\pi\right)\right)$$

Equation 2.3: Back-EMFcoefficient as a function of the angular position of the motor

where $\zeta = \frac{x_0}{g_0}$ (see 3.6) and k_e^i is the trapezoidal wave-shaped normalized back-EMF relative to the i^{th} phase of the non-damaged model.

2. Three-phase RL model block: it receives in input the commanded voltages and the normalized coefficients calculated from the output of the previous block.

   The normalized back-EMF coefficient is the ratio between back-EMF and the angular speed of the motor ω_n.

$$k_{bemf} = \frac{BEMF}{\omega_m} = BEMF_{norm}$$

Equation 2.4: Back-EMFcoefficient

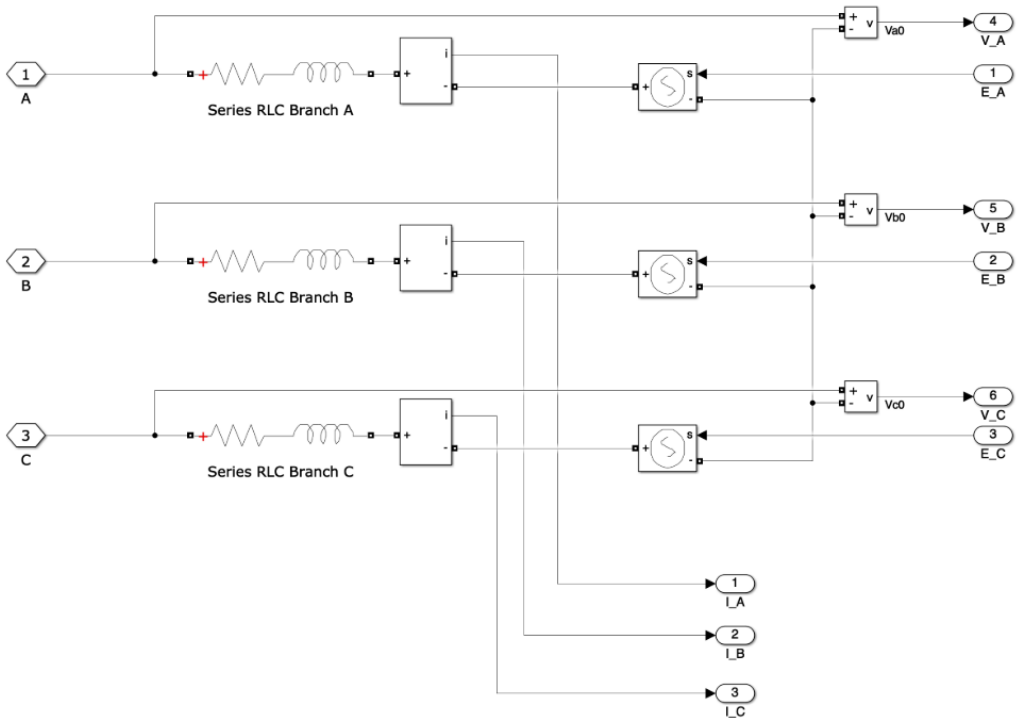


Figure 16: Three-phase RL model block

The block contains some components of the Simscape library that must be converted before use in Simulink.

The block outputs are the voltage drops across the phases and each phase current.

3. Computation of motor torque block: it calculates the motor torque starting from the outputs of the two previous blocks because it exploits the back-EMF coefficients of the first block and the currents of each phase of the second block.

   These parameters are multiplied respectively for each phase, added together and finally passed through a torque saturation block.

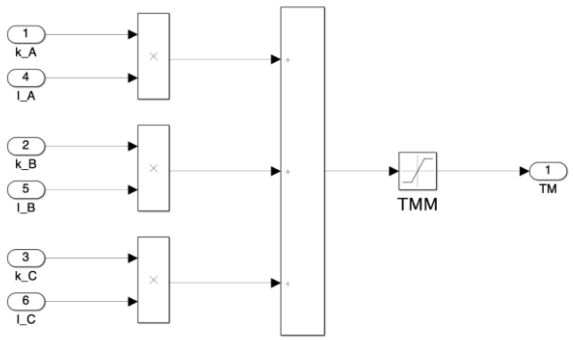


Figure 17: Computation of motor torque block

2.3.5. Motor-transmission dynamical subsystem

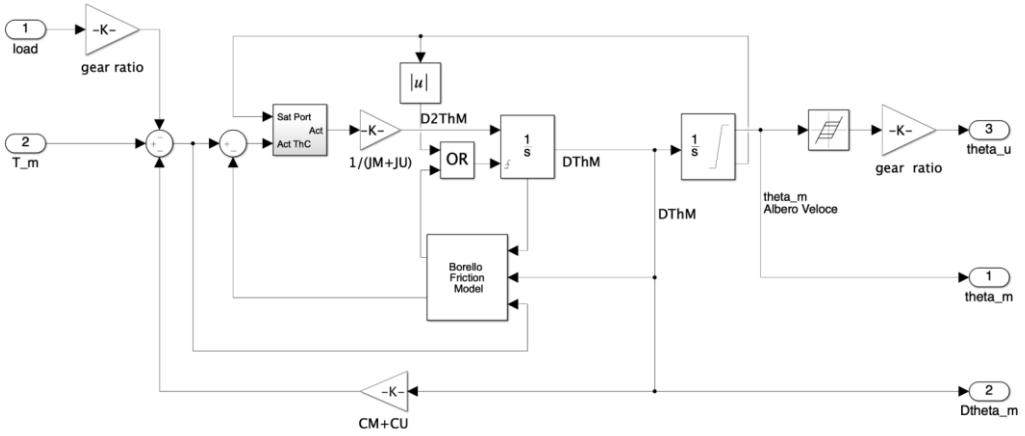


Figure 18: Motor-transmission dynamical subsystem

In this block, the mechanical transmission has been modeled which allows passing from low torque and high angular speed (fast shaft) to high torque and low angular speed (slow shaft).

The gear ratio is the most important parameter of the subsystem. The Borello model is used for friction modeling. For a correct representation of the phenomenon, the saturation port is used, which allows avoiding the limit cycles. In addition, end-stops are considered through the saturation blocks present on Simulink.

The system evolves with a dynamic of the first order as the feedback loop is made on the speed and not on the position of the motor.

The block outputs the angular position of the user, which is the one we want to control through the PID controller, the motor speed and the angular position of the motor. The latter is read by the Hall sensors and used to calculate the back-electromotive force and the switching logic of the phases.

2.4. F-16 Longitudinal dynamics subsystem

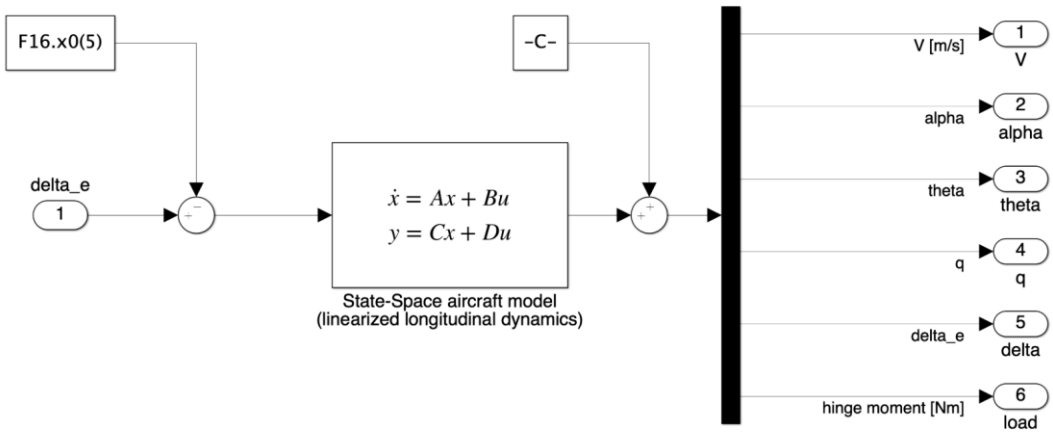


Figure 19: F16 Longitudinal dynamics subsystem

This block receives the deflection of the elevator as input and using the state-space model of the aircraft calculate the vector of states and more.

This model simulates the response of the entire aircraft only in the longitudinal plane, so the command vector is formed only by the variable δ_e.

$F16.x0(5)$ represents the initian condition of elevator's deflection.

The subsystem outputs the velocity of the aircraft V, angle of attack α, angle of pitch θ (Euler's angle), pitch rate q, elevator's deflection δ_e and hinge moment H.

All the matrices that populate the model are the same as those mentioned in the book "Aircraft Control and Simulation". [5]

3. Fault Modes

EMAs are mainly used for the control of secondary flight surfaces (trim-tab, spoilers, speedbrakes) in civil aircraft and military aircraft. However, engineers are studying these devices to adopt them also for primary flight controls and, since EMAs are a recent technology when compared to electrohydraulics and hydromechanics, it is very important to be aware of the fault modes of these actuators [6]. Most of this chapter is based on [2].

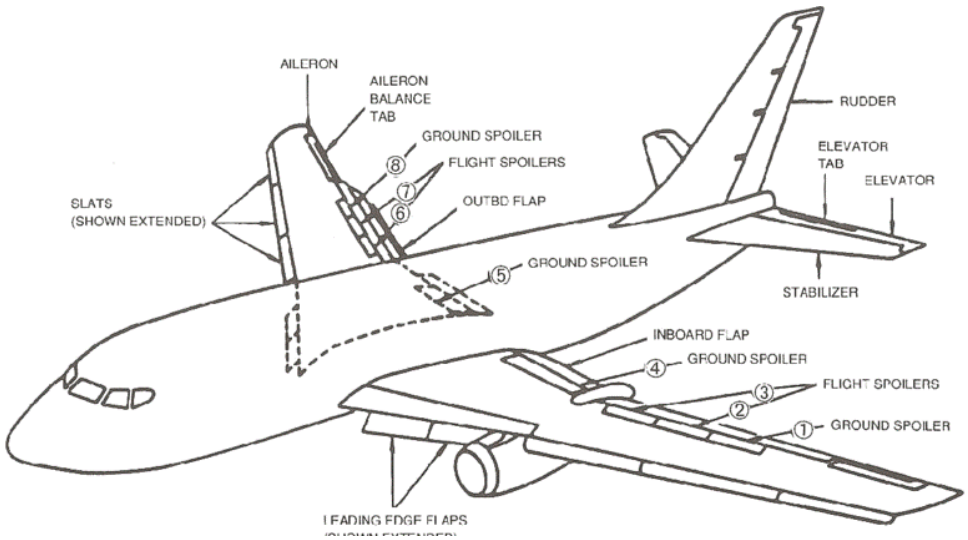


Figure 20: Flight Control Surfaces of Jet Passenger Carrier [7]

3.1. Definition

First, we need to explain the difference between fault and failure:

- Fault: state of an item characterized by inability to perform as required.
- Failure: the event resulting in an item being no longer able to perform its required function. [8]

Basically, the failure is the event while the fault is the state. In aeronautics it is very common to talk about Fault Tree Analysis [FTA] and Failure Modes Effects (and

Criticality) Analysis [FMEA/FMECA]. These methods all have the objective of identifying risk factors, the possible causes of all undesirable events that can influence mission reliability. The FTA can understand and identify the causes of a certain undesirable event, evaluate human errors and quantify the probability of failure. Furthermore, it allows for understanding the interactions between the different faults.

There are standards for system security, grouped in MIL-STD-882. Based on the severity of the consequences, the legislation performs a classification of faults.

Categories	Consequences	Description
I	Catastrophic	A failure which may cause death or weapon system loss
II	Critical	A failure which may cause severe injury, major property damage, or major system damage which will result in mission loss
III	Marginal	A failure which may cause minor injury, minor property damage, or minor system damage which will results in delay or loss of availability or mission degradation
IV	Negligible	A failure not serious enough to cause injury, property damage, or system damage, but which will result in unscheduled maintenance or repair

Table 1: Failure severity classification (MIL-STD-882)

Severity is usually linked to the probability of an event occurring. The result is a risk assessment matrix.

RISK ASSESSMENT MATRIX						
		Probability				
Severity		Frequent A	Likely B	Occasional C	Seldom D	Unlikely E
Catastrophic	I	E	E	H	H	M
Critical	II	E	H	H	M	L
Marginal	III	H	M	M	L	L
Negligible	IV	M	L	L	L	L
E – Extremely High		H – High		M – Moderate		L – Low

Table 2: Risk Assessment Matrix (MIL-STD-882E)

In a system, it is good practice to arrange multiple elements so that it has greater resistance to a fault. Redundancies must be designed to avoid common failure modes.

3.2. EMAs Fault Modes

An EMA can be affected by 4 types of fault modes:

Motor Faults:	The BLDC motor that is used in the EMA presents the problem of not having any type of active cooling and for this reason, heat management is a problem to be addressed. If the temperature inside the motor is very high, the insulation between the stator coils can degrade causing a short circuit that can irremediably damage the device.

	Furthermore, if the Curie temperature is exceeded, a ferromagnetic material loses its magnetic characteristics (for example the rotor can become demagnetized). Additionally, it operates at very high rotation speeds and for this reason vibrations are induced on the rotor bearings and high inertial loads.
Electrical and electronic faults:	These are faults affecting the electrical power system and the control system: short circuits, overheating, degradation of the welds, degradation of the wires, degradation of the connectors, degradation of the insulation, overvoltages and overcurrents.
Mechanical faults:	Due to the application of excessive aerodynamic loads, the transmission or the moving surface can deform and therefore induce a malfunction of the system. Even improper maintenance of the system, poor lubrication (with a consequent increase in heat due to friction and the onset of electrical and motor faults); the propagation of a crack can lead to the total breakage of a component. Often even the environment in which the device is found to operate can be decisive: just think of phenomena such as galvanic corrosion.
Sensor faults:	The loss of the signal coming from the Hall sensors can mean the loss of the actuator because it is no longer able to manage the switching logic for the excitation of the phases. There can be three types of faults: bias, drift and scaling.

3.3. Friction fault

Friction is the force resisting the relative movement of two bodies sliding against each other. On the surfaces of all objects, there are tiny bumps and ridges. Those

microscopic peaks and valleys catch on one another when two objects are moving past each other.

The Stribeck curve is an overall view of friction variation forthe entire transition of lubrication modes, from the boundary and mixed lubrication up to the full-film hydrodynamic of static friction.

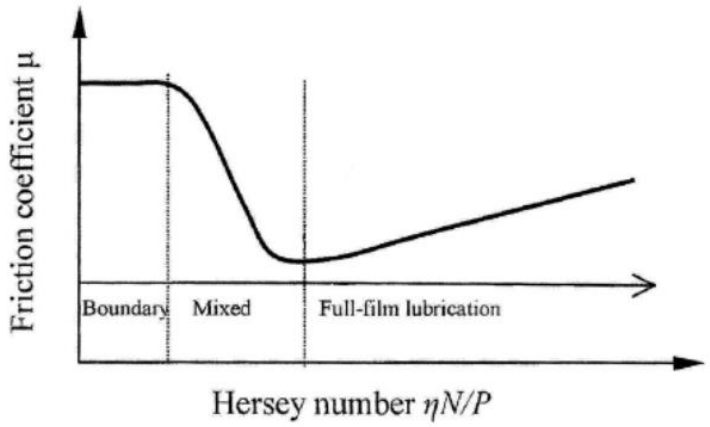


Figure 21: Stribeck curve (Hersey number is the relative velocity between the contact surfaces) [9]

In the boundary layer region, the friction force is high, approximately constant and is not affected by external load and speed.

In the mixed lubrication regime, i.e. when the speed begins to increase, the friction force drops sharply as lubricating fluid (air) is pushed between the contact surfaces.

As the speed increases, we enter the full-film lubrication regime, the friction force begins to increase again as the shear strain rate increases.

3.3.1. Model

The problem of dynamic modeling (obtained by numerical simulation) of the dry friction is that the calculation procedure does not notice the speed passing through zero and, therefore, does not carry out any checks regarding the possible stopping of the mechanical organ. If this happens, by integrating the acceleration between the start and end of the step, two-speed values of opposite sign are obtained. This

check must instead be conducted because, in stationary conditions, the friction force (or torque) may be greater than the active one.

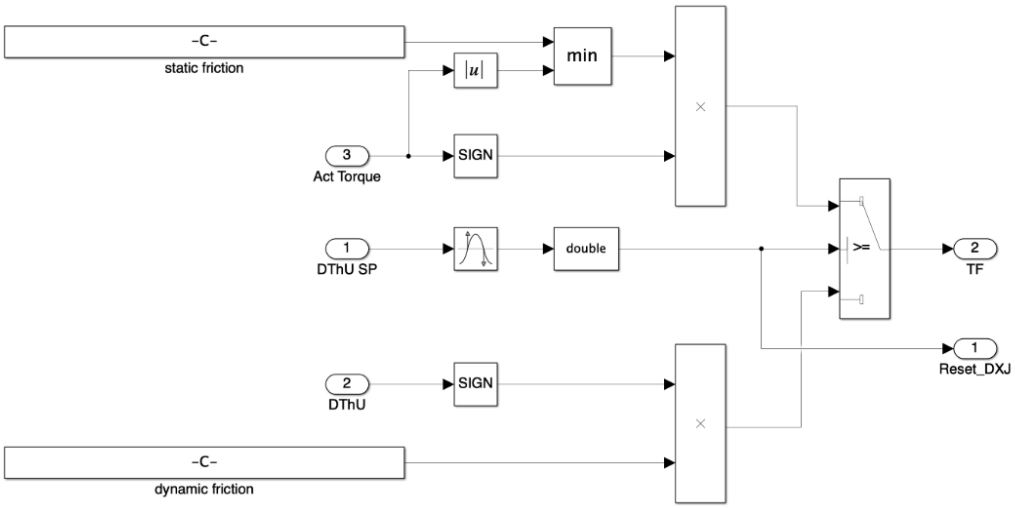


Figure 22: Borello friction model implementation in Simulink

The Borello model [10] is an evolution of the Coulomb model which has several advantages:

- It distinguishes the sign of the frictional torque as a function of the direction of the speed.
- It distinguishes the adhesion conditions from the dynamic ones (in fact, two distinct values can be assigned for the torque - friction, F_{SJ} in static or grip conditions and F_{DJ} in dynamic conditions).
- It evaluates the possible stop of the mechanical element initially in motion.
- It keeps the mechanical element correctly stationary (or in motion) in adherence conditions (or motion).
- It evaluates the possible restart of the initially stopped mechanical element.
- It considers, in a single model, the presence of end-stop.
- It does not need a dimensionless ϵ parameter which is very difficult to determine, unlike friction models such as Karnopp's model and Quinn's model.

The mathematical model is the following:

$$TF = \begin{cases} T_{act}\,, & if\ v = 0 \wedge |F_{act}| \leq T_{SJ} \\ sgn(T_{act}) \cdot T_{SJ}, & if\ v = 0 \wedge |F_{act}| > T_{SJ} \\ sgn(v) \cdot T_{DJ}, & if\ v \neq 0 \end{cases}$$

Equation 3.1: Borello mathematical model for torque friction

where, T_{act} is the torque acting on the system, T_{SJ} is the static friction force, T_{DJ} is the dynamic friction force and v is the relative speed between the two surfaces.

In the model, there is a zero-crossing detection block. This block outputs one when it detects the speed signal passing through zero (from less to greater than zero or vice versa). When this happens, the speed is set to zero and the static friction force is set.

Algebric loops are avoided by using the state port.

3.4. Noise fault

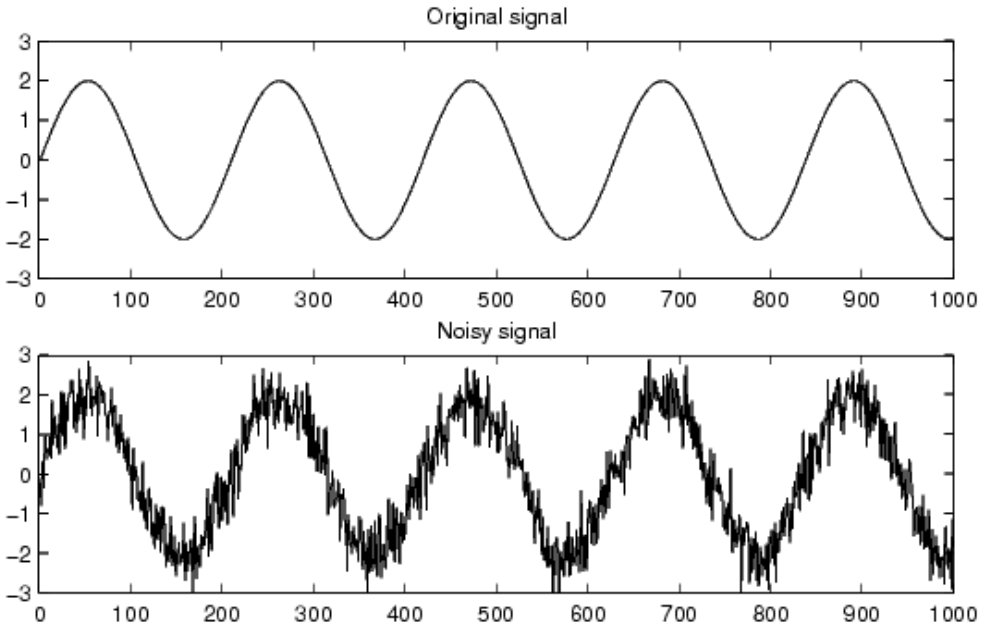


Figure 23: Noise signal [11]

Noise can be defined as an unwanted signal that interferes with the communication or measurement of another signal. A noise is a signal that carries information about the sources of the noise itself and the environment in which it propagates. [12]

The sources of noise include:

- Electronic noise:
    - Thermal noise generated by the random movements of thermally energised particles in a conductor.
    - Shot noise i.e. fluctuations of electrons.
    - Burst noise caused by step transitions of as high as several hundred millivolts, at random times and durations.
- Acoustic noise: revolving machines, computer fans, moving vehicles, people talking in the background, wind and rain produce vibrations.
- Electromagnetic noise: electromagnetic induction and conduction due to atmosphere.
- Electrostatic noise: presence of voltage with or without current flow i.e. fluorescent lighting.
- Processing noise: lost data packets in digital communication systems.
- CO-Channel noise: crosstalk from two different radio transmitters on the same frequency channel.

Depending on its frequency spectrum or time characteristics, a noise process can be further classified into one of several categories. When we speak of white noise, for example, we refer to purely random noise that has an impulse autocorrelation function and a flat power spectrum. It contains equal power at all frequencies.

The autocorrelation function r_{nn} of a continuous-time zero-mean white noise process, $n(t)$, is given by:

$$r_{nn}(\tau) = E[n(t)n(t+\tau)] = \sigma_n^2\,\delta(\tau)$$

Equation 3.2: Autocorrelation function of pure white noise

where $\sigma_n{}^2$ variance is a particular delta function.

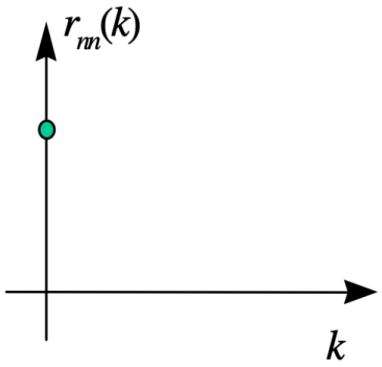


Figure 24: Autocorrelation function of pure white noise

Using the Fourier transform of the autocorrelation function, the power spectrum of white noise is obtained:

$$P_{nn}(f) = \int_{-\infty}^{\infty} r_{nn}(t) e^{-j2\pi f t}\, dt = \sigma_n^2$$

Equation 3.3: Power spectrum of pure white noise

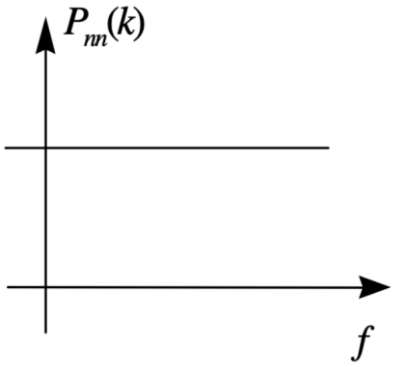


Figure 25: Power spectrum of pure white noise

Since a pure white noise need to have infinite power to cover an infinite range of frequencies, it is a theoretical concept. A more practical concept is band-limited white noise, defined as a noise with a flat spectrum in a limited bandwidth B:

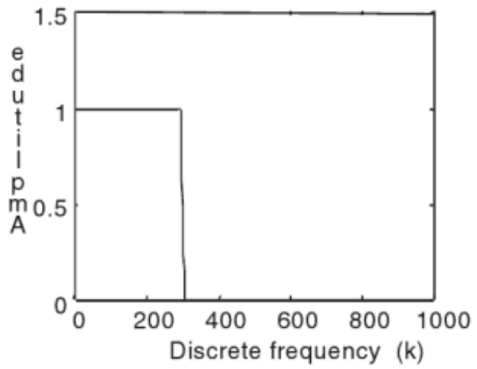


Figure 26: Power spectrum of band-limited white noise

$$P_{NN} = \begin{cases} \sigma^2 & if\ |f| \leq B \\ 0 & if\ |f| > B \end{cases}$$

Equation 3.4: Power spectrum of band-limited white noise

The autocorrelation function of a band-limited white noise, in case of discrete-time, has the shape of a sinc function:

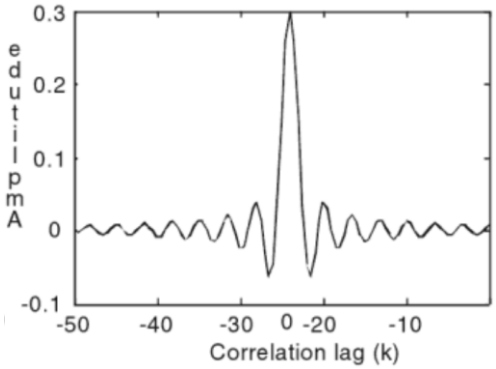


Figure 27: Autocorrelation function of band-limited white noise

$$r_{nn}(T_s k) = 2B\sigma_n^2 \frac{sin(2\pi B T_s k)}{2\pi B T_s k}$$

Equation 3.5: Autocorrelation function of band-limited white noise

where T_s, is the sampling period.

When $T_s = \frac{1}{2B}$, i.e. when the sampling rate is equal to the Nyquist rate, the equations becomes:

$$r_{NN}(T_s k) = 2B\sigma_n^2 \frac{sin(\pi k)}{\pi k} = 2B\sigma_n^2 \delta(k)$$

Equation 3.6: Autocorrelation function of band-limited white noise when $T_S = 1/2B$

3.4.1. Model

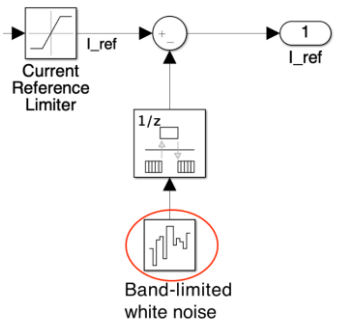


Figure 28: Band-limited white noise in Control Electronics (PID) subsystem

A band-limited white noise model is implemented in the Control Electronics (PID) subsystem superimposing it on the current reference signal. The highest noise frequency is set at less than half of the sampling rate.

3.5. Short circuit fault

When the motor operates at a very high temperature, the insulation covering the wires that make up the stator coils degrades and allows the formation of short circuits.

The formation of a short circuit can take place:

- between two wires of the same phase (coil-coil): the resistance and inductance of the motor decrease; consequently, the current increases and

the motor is brought to overheating. This condition is the first that is observed.

- between two wires of different phases (phase-phase): generally, this condition occurs after the coil-coil and leads to a fault condition.
- between a wire and the stator iron core (phase-ground): generally, this condition occurs after the coil-coil and leads to a fault condition.

3.5.1. Model

As just mentioned, a short circuit between two wires leads to a decrease in resistance and inductance proportional to the number of shorted wires. Consequently, one effect is the reduction of the back-electromotive force which can be expressed in the following way:

$$\boldsymbol{k_{bemf} = G_M = \frac{\partial \phi}{\partial \theta_m} = nA\frac{\partial}{\partial \theta_m}\left(\int_A \boldsymbol{B} \cdot \boldsymbol{n}\, \boldsymbol{dS}\right)}$$

Equation 3.7: BEMF reduction due to short circuit

where A is the total winding area, **B** the magnetic flux density of the rotor and n is the number of windigs making up the coil.

Defining N_i as the normalized value of shorted coil windings in respect to n, we can express:

$$R_i = N_i R$$
$$L_i = N_i^2 L$$
$$K_e^i = N_i k_e$$

where R_i, L_i and k_e^i are respectively resistance, inductance and normalized BEMF coefficient of the i^{th} phase while R, L and k_e are the nominal values, referring to a zero-fault condition.

3.6. Rotor eccentricity fault

It is possible to distinguish two types of rotor eccentricity:

Type	Definition	Effects
Static eccentricity	The misalignment error between the rotor rotation axis and the symmetry axis	Irregularities in the air gap surrounding the rotor (different for each phases)
Dinamic eccentricity	The misalignment error between the rotor rotation axis and its rotational inertia axis	Vibrations, bearing wear, non-constant torque

Table 3: Static and Dinamic rotor eccentricity fault differences

The effects of static eccentricity are now analyzed, i.e. the irregularities in the air gap surrounding the rotor. The rotor is assumed to be a rigid body without deformations.

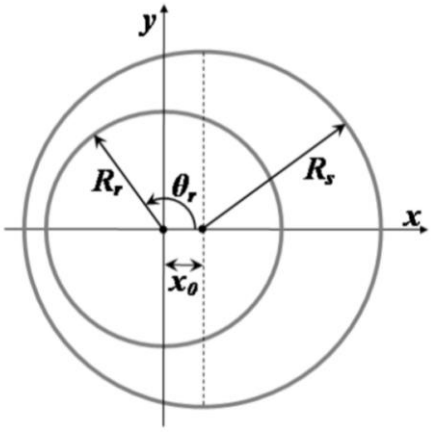


Figure 29: Air gap surrounding the rotor

From the figure we extrapolate the following relationships:

Cartesian Coordinates	Polar Coordinates
$x^2 + y^2 = R_r^2$	$\rho = R_r$
$(x - x_0)^2 + y^2 = R_s^2$	$\rho^2 - 2\rho\, cos\theta + x_0^2 = R_s^2$

Table 4: Stator and Rotor equation with static eccentricity

where x_0 is the distance between the two axes, R_r is the rotor radius and R_s is the stator radius.

Now we define the air gap g:

$$g = x_0\, cos\theta + R_s\sqrt{1 - \left(\frac{x_0}{R_s}\right)^2 sin^2\theta} - R_r \approx$$

$$\approx x_0\, cos\theta + R_s\left(1 - \frac{1}{2}\left(\frac{x_0}{R_s}\right)^2 sin^2\theta\right) - R_r \approx$$

$$\approx x_0\, cos\theta + g_0$$

Equation 3.8

where $g_0 = R_s - R_r$ is the air gap considered during nominal condition.

A new parameter, called eccentricity parameter ξ, can now be entered:

$$\xi = \frac{x_0}{g_0}$$

Equation 3.9: Eccentricity Parameter

$$g \approx g_0(1 + \xi\, cos\theta)$$

Equation 3.10: Air gap approximation

The magnetic flux Φ calculated solving the circuit between two consecutive rotor poles is:

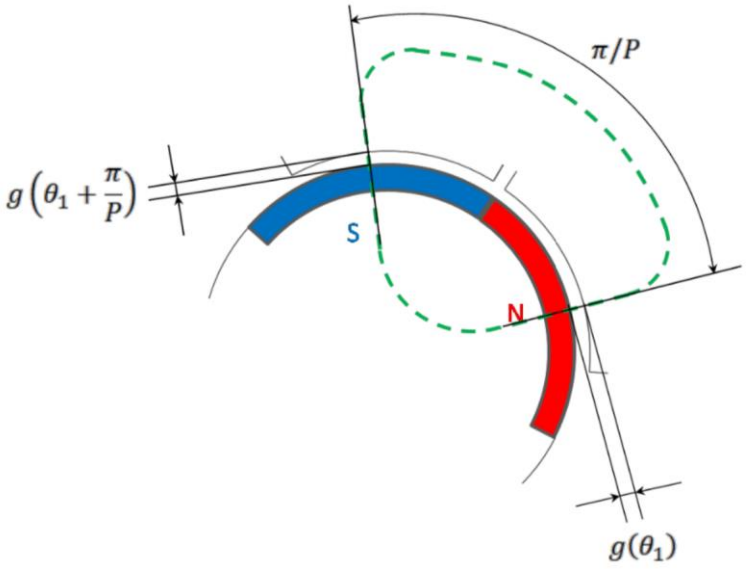


Figure 30: Air gap approximation

$$\Phi = \frac{F_m}{\frac{g(\theta_1)}{\mu_0 S} + \frac{g\left(\theta_1 + \frac{\pi}{P}\right)}{\mu_0 S}} = \frac{F_m \mu_0 S}{g(\theta_1) g\left(\frac{\pi}{P}\right)} = \frac{2\Phi_0 g_0}{g(\theta_1) g\left(\frac{\pi}{P}\right)}$$

Equation 3.11: Magnetic flux

where F_m is the rotor magneto-motive force, P is the number of poles, S is the surface crossed by magnetic flux and Φ_0 is the nominal flux.

The change in magnetic flux over time is the back-ElectroMotive Force.

3.6.1. Model

As shown in [13] the rotor eccentricity fault can be described through a block diagram without involving the FEM analysis.

We can express the back-ElectroMotive Force through the following relationship:

$$k_{bemf}^{i} = k_{e}^{i}(\theta_m) \cdot \left(1 + \xi \, cos\left(\theta_m + \frac{2(i-1)}{3}\pi\right)\right)$$

Equation 3.12: Back-electromotive force coefficient

where $k_{e}^{i}(\theta_m)$ is the non-faulty condition traperzoidal-shaped coefficient.

4. Artificial Neural Network

The neural network is one of the most interesting programming tools in this historical period. Mankind is trying to make machines have their own intelligence and for this reason, neural networks play a fundamental role. In a neural network, the programmer does not tell the computer how to solve problems. It, by contrast, learns itself from a dataset the relationship between input and output. These techniques are used extensively in computer vision and speech recognition. They are also studied at a company level by Google and Facebook. Our goal is to implement them within an electromechanical servomechanism.

4.1. Concepts

Before talking about the architecture of a neural network, it is important to clarify the concept of perceptron, neuron, activation function, weights, and bias.

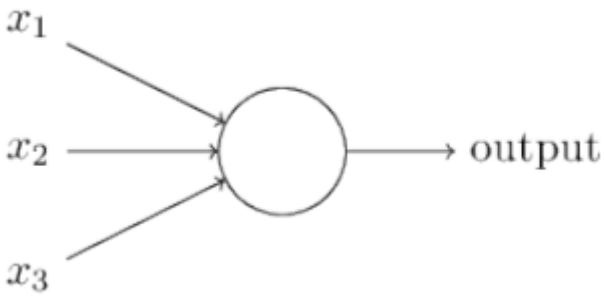


Figure 31: Perceptron

A perceptron takes several binary inputs and produces a single binary output. The scientist Frank Rosenblatt introduced weights to compute the output. These elements are real numbers, and they express the value of their input to the output.

Instead the output is calculated based on the value of the sum $\sum_j \theta_j x_j$.

If the latter is greater than a threshold value, the output is 1 and vice-versa. Formally:

$$\mathbf{output} = \begin{cases} \mathbf{0} & \boldsymbol{if} \sum_j \boldsymbol{\theta}_j \boldsymbol{x}_j \leq \boldsymbol{threshold} \\ \mathbf{1} & \boldsymbol{if} \sum_j \boldsymbol{\theta}_j \boldsymbol{x}_j > \boldsymbol{threshold} \end{cases}$$

Equation 4.1

If $b \equiv -threshold$, we can rewrite the relationship as it follows:

$$\boldsymbol{output} = \begin{cases} \mathbf{0} & \boldsymbol{if} \sum_j \boldsymbol{\theta}_j \boldsymbol{x}_j + \boldsymbol{b} \leq \mathbf{0} \\ \mathbf{1} & \boldsymbol{if} \sum_j \boldsymbol{\theta}_j \boldsymbol{x}_j + \boldsymbol{b} > \mathbf{0} \end{cases}$$

Equation 4.2

"The perceptron's bias (b) is a measure of how easy it is to get the perceptron to output 1". [14]

Another type of neuron is the sigmoidal neuron that instead of responding with an output of only 0 and 1 can return a value within the range.

The letter σ indicates the sigmoid function, which is very similar to the step function but smoother:

$$\boldsymbol{\sigma(z)} = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e^{-z}}}$$

Equation 4.3: The sigmoid function

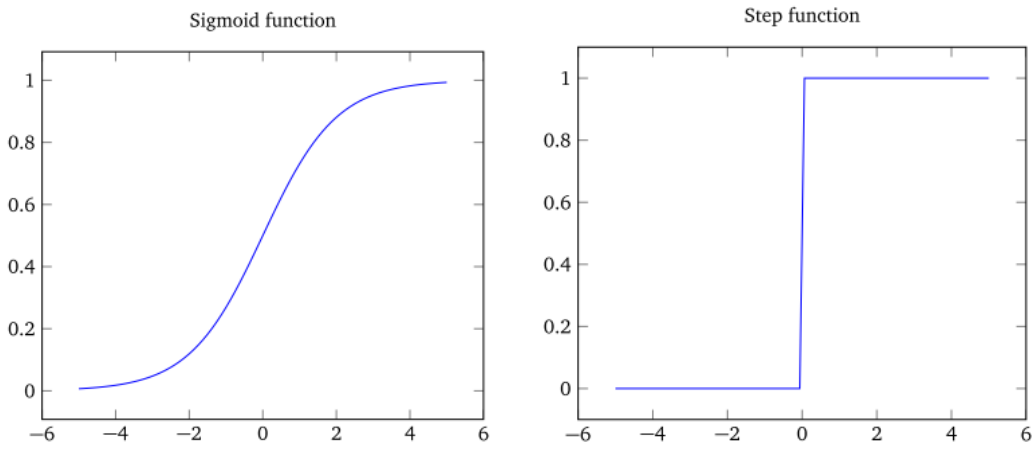


Figure 32: Comparison between sigmoid function and step function

4.2. The architeture of neural networks

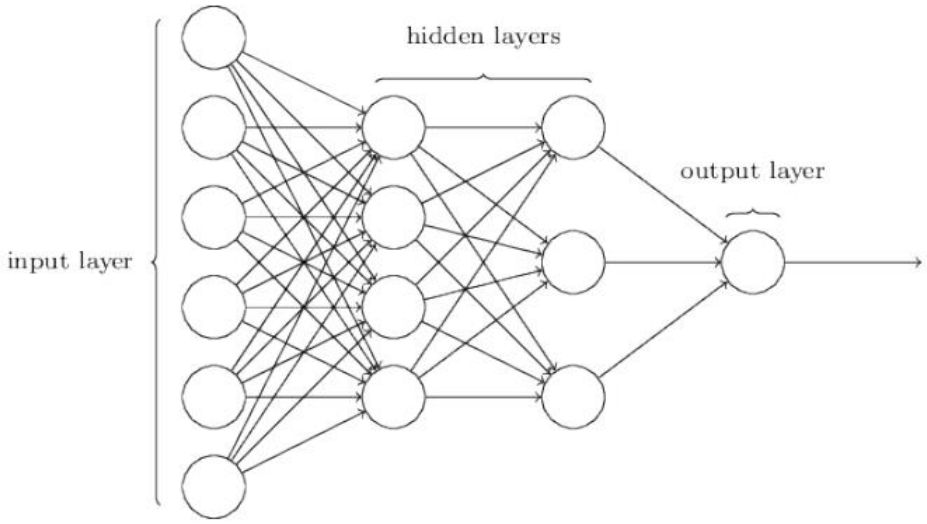


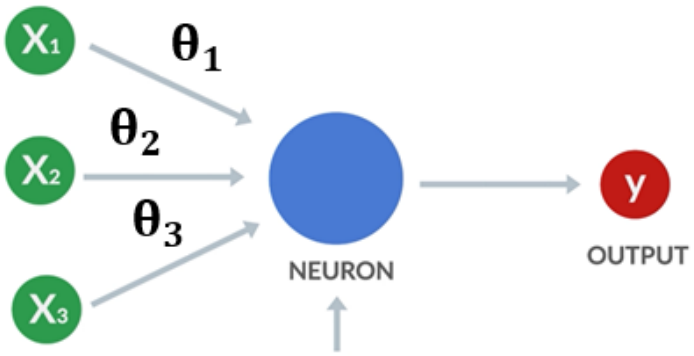
Figure 33: The architeture of neural network

Simple neural networks are basically composed of three layers. Each column of neurons represents a layer while each arrow represents a weight. The leftmost layer is called the input layer and the rightmost is called output layer. The middle layer is called hidden layer, since the neurons in this layer are neither inputs nor outputs.

Generally, there are multiple hidden layers in a network and when this happens, we are talking about of a *deep* neural network (for example, in the figure above the net has two hidden layers). The output of a previous layer is the input of the next layer and this is the main feature of the feedforward network.

4.3. The Activation Function

Inside the neuron, the inputs are multiplied by their respective weights and then added together by adding the bias value. The obtained value represents the activation input $z = f(x, \theta, b)$. The latter must go through an activation function which is responsible for establishing whether the activation input is intense enough to be able to be propagated to the next neuron and then activate it. [15]

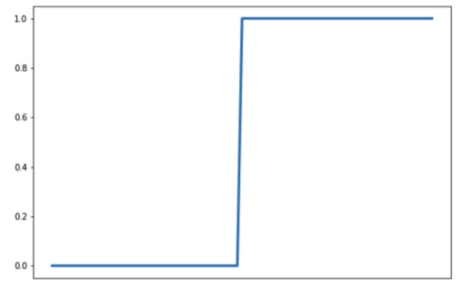


$$z = x_1\theta_1 + x_2\theta_2 + x_3\theta_3 + b$$

$$h_\theta(x) = \phi(z)$$

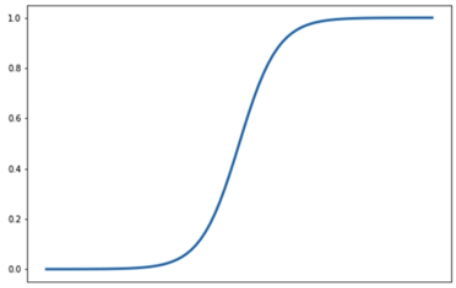
Figure 34: The Activation Funtion

Step



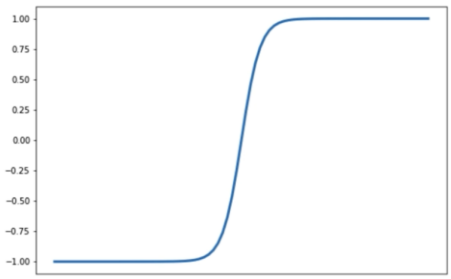
$$\phi(z) = \begin{cases} 0 & if\ z \leq 0 \\ 1 & if\ z > 0 \end{cases}$$

Sigmoid



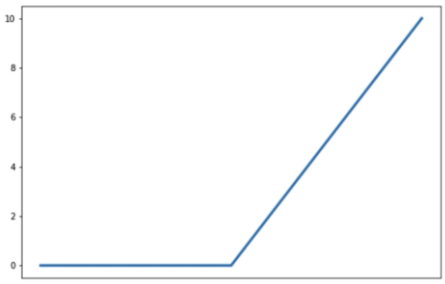
$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Tanh



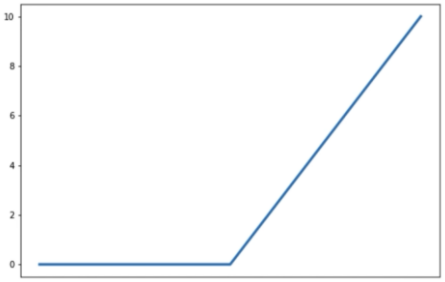
$$\phi(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}}$$

Rectified Linear Unit (ReLU)



$$\phi(z) = \begin{cases} 0 & if\ z \leq 0 \\ z & if\ z > 0 \end{cases}$$

Leacky ReLU



$$\phi(z) = \begin{cases} 0.01z & if\ z \leq 0 \\ z & if\ z > 0 \end{cases}$$

Using the step function, the quantitative information related to the activation input is lost: an activation input of 0.5 will return 1 as well as an activation input of 1000. For this reason, the step function is never used. Generally, the activation functions used in the hidden layers are the ReLU and the tanh. In the last layer, the sigmoid function or the linear function are used depending on whether it is a classification or regression problem.

4.4. Learning

To train the neural network, we need to give it the inputs from our dataset and compare its hypothesis outputs $h_\theta(x)$ with the outputs y from the dataset.

	Features 1		Features N	Output 1		Output N
Example 1	x_{11}	…	x_{1N}	y_{11}	…	y_{1N}
Example 2	x_{21}	…	x_{2N}	y_{21}	…	y_{2N}
…	…	…	…	…	…	…
Example M	x_{M1}	…	x_{MN}	y_{M1}	…	y_{MN}

Table 5: Input and Output of a neural network

From this, we understand that it is important to have a large dataset and a large amount of computational power. Since the neural network is still untrained, its hypothesis outputs will be wrong. In the next paragraphs we define a cost function that shows us how much the neural network's outputs are wrong from the real outputs and how to train them.

4.5. Multivariate Linear Regression

A multivariate linear regression is a linear regression with multiple variables. For simplicity, only one output is considered.

Let us imagine we have a dataset structured as follows:

	Features 1 $\boldsymbol{x_1}$	**Features 2** $\boldsymbol{x_2}$		**Features n** $\boldsymbol{x_n}$	**Output** $\boldsymbol{y}$
Example 1 $\boldsymbol{x^1}$	x_1^1	x_2^1	…	x_n^1	y_1
Example 2 $\boldsymbol{x^2}$	x_1^2	x_2^2	…	x_n^2	y_2

…	…	…	…	…	…
Example m $\boldsymbol{x^m}$	x_1^m	x_2^m	…	x_n^m	y^m

Notation:

- n: number of features.
- m: number of training examples.
- $x^{(i)}$: input (features) of i^{th} training example.
- $x_j{}^{(i)}$: value of feature j in i^{th} training example.

The hypothesis function that links each features of an example to the output is the following:

$$\boldsymbol{h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n}$$

Equation 4.4: The hypothesis function

For convenience reasons we assume $x_0^{(i)} = 1$ for $i \in [1, m]$.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_0} \\ \boldsymbol{x_1} \\ \cdots \\ \boldsymbol{x_n} \end{bmatrix} \boldsymbol{\epsilon} \mathbb{R}^{\boldsymbol{n+1}} \qquad \boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta_0} \\ \boldsymbol{\theta_1} \\ \cdots \\ \boldsymbol{\theta_n} \end{bmatrix} \boldsymbol{\epsilon} \mathbb{R}^{\boldsymbol{n+1}}$$

Equation 4.5: Input vector and weight vector

$$\boldsymbol{h_\theta(x)} = [\boldsymbol{\theta_0} \quad \boldsymbol{\theta_1} \quad \ldots \quad \boldsymbol{\theta_n}] \begin{bmatrix} \boldsymbol{x_0} \\ \boldsymbol{x_1} \\ \cdots \\ \boldsymbol{x_n} \end{bmatrix} = \boldsymbol{\theta^T x}$$

Equation 4.6: The hypothesis function vector shape

4.6. Cost Function (mse)

The cost function expresses the accuracy of our hypothesis function with respect to the real value of the output. In fact, for a linear regression it can be expressed as:

$$\boldsymbol{J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2}$$

Equation 4.7: Cost Function (mse)

Where $h_\theta(x_i)$ is the predicted value (hypothesis function), y_i is the actual value and m is the number of examples. This function is also called "mean squared error" (mse) because the difference between them represents an error, which is squared and finally all the errors of each single example are added together and divided by the number of examples (mean).

4.6.1. Gradient Descent

Since the value of $h_\theta(x_i)$ is a function of the weights θ and of the biases b we want to find those values that best approximate the predicted function to the real function, i.e. we want the cost function to be as small as possible.

We are helped by the gradient descent because its task is precisely this, that is to update the value of weights and bias in such a way as to minimize the cost function.

The gradient descent algorithm repeats the following rule until convergence:

$$\boldsymbol{\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)}$$

Equation 4.8: Gradient descent algorithm

Where θ_j represents the j^{th} weight (the same is true for biases).

The hyperparameter α is known as the learning rate. It allows us to vary the size of each single step: the higher this parameter, the larger the step. The direction in which the step is taken is determined by the partial derivative of $J(\theta)$ the minus sign has the task of inverting the trend of the gradient.

Substituting the expression of h_θ in the case of linear regression for a single example we obtain:

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{\partial}{\partial\theta_j}\left(\frac{1}{2}(h_\theta(x)-y)^2\right) = 2\cdot\frac{1}{2}(h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}(h_\theta(x)-y)$$

$$= (h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right) = (h_\theta(x)-y)x_j$$

Equation 4.9: Partial derivative of $J(\theta)$ in the case of linear regression for a single

In the case of multiple variables, the gradient descent algorithm takes the following form:

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i)-y^i)\cdot x_j^i \quad for\ j := 0\ldots.n$$

Equation 4.10: Gradient descent algorithm in the case of multiple variables

The choice of the learning rate α is very important because:

- A very small value leads to slow convergence.
- A very large value can lead to divergence (bounce from one side of the curve to the other without ever reaching the minimum).

Since two features can have very different ranges of values, the speed of the grandient descent can differ and oscillate inefficiently when the variables are irregular.

For this reason, the feature scaling technique is carried out, i.e. all the features are brought into the same range, for example (0,1) thanks to the following conversion formula:

$$x_i := \frac{x_i - min(x_i)}{\max(x_i) - \min(x_i)}$$

Equation 4.11: Feature scaling

A very similar method is mean normalization, in which the average value is taken as a reference:

$$x_i := \frac{x_i - average(x_i)}{\max(x_i) - \min(x_i)}$$

Equation 4.12: Mean normalization

4.6.2. Backpropagation

The backpropagation is the algorithm that allows minimizing the cost function in an artificial neural network, the same task performed by the gradient descent in linear regression. This process occurs through the determination of the derivative of the cost function.

Let us imagine we have a dataset and one training example (x, y). [16]

Forward propagation consists in the calculation of:

$$\begin{aligned}
a^{(1)} &= x \\
z^{(2)} &= \Theta^{(1)} a^{(1)} \\
a^{(2)} &= g(z^{(2)}) \quad (\text{add } a_0^{(2)}) \\
z^{(3)} &= \Theta^{(2)} a^{(2)} \\
a^{(3)} &= g(z^{(3)}) \quad (\text{add } a_0^{(3)}) \\
z^{(4)} &= \Theta^{(3)} a^{(3)} \\
a^{(4)} &= h_\Theta(x) = g(z^{(4)})
\end{aligned}$$

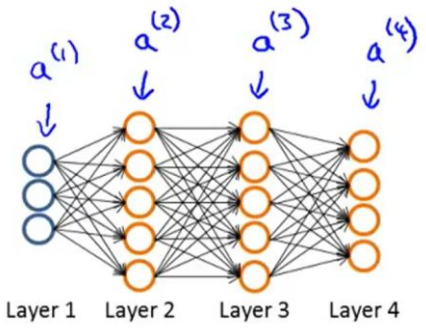


Figure 35: Forward propagation

where $a^{(i)}$ is the activation values of i^{th} layer i.e the value of z to which the activation function g is applied. $a^{(4)}$ is the last activation value and coincides with the hypothesis value $h_\Theta(x)$.

Remembering that $a_j^{(l)}$ represents the activation values of the node j in layer l, the intuition of the backpropagation algorithm is the calculation of error values of node j in layer l called $\delta_j^{(l)}$.

The delta term captures the error in the activation of that node, for example:

$$\delta_j^{(L)} = a_j^{(L)} - y_j$$

Equation 4.13: Delta last layer error values

where L is the total number of layes. In vector form:

$$\delta^{(L)} = a^{(L)} - y = h_\Theta(x) - y$$

Equation 4.14: Delta last layer error vectorial form

Now let us proceed with the calculation of $\delta^{(L-1)}$ back to $\delta^{(2)}$, hence the name backpropagation ($\delta^{(1)} = 0$, since $a^{(1)}$ corresponds to the features).

$$\delta^{(l)} = \left(\left(\Theta^{(l)}\right)^T \delta^{(l+1)}\right) .* \; g'\left(z^{(l)}\right)$$

Equation 4.15: Delta error in previous layers

where $g'\left(z^{(l)}\right) \cong a^{(l)} .* \left(1 - a(l)\right)$ is the derivative of the activation function g evaluated with the input values given by activation input $z^{(l)}$, $\Theta^{(l)}$ is the matrix of weights of l layer and the symbol $.*$ identifies the element-wise multiplication.

Now we can determine and update the Δ matrix values at each iteration:

$$\Delta_{ij}^{(l)} \coloneqq \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

Equation 4.16: Delta matrix values

Finally, the last matrix D is calculated, which coincides with the derivative of the cost function $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$ as a function of each weight:

$$D_{ij}^{(l)} \coloneqq \begin{cases} \frac{1}{m}\left(\Delta_{ij}^{(l)} + \lambda \Theta_{\mathrm{ij}}^{(l)}\right) & if\ j \neq 0 \\ \frac{1}{m}\Delta_{ij}^{(l)} & if\ j = 0 \end{cases}$$

where $j = 0$ coincides with the case of bias.

4.6.3. Underfitting and Overfitting

The training error tends to decrease as the degree d of the polynomial of the function $h_\theta(x)$ increases as we are trying to adopt a more complex model. [16]

Similarly, the cross-validation error tends to decrease as d increases, however at a certain point the trend assumes an opposite behavior, forming a convex curve.

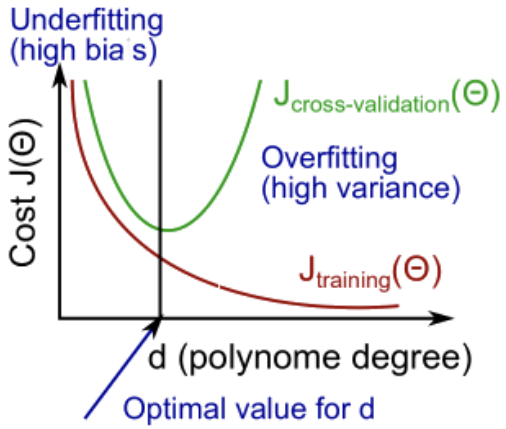


Figure 36: Underfitting vs Overfitting

So we talk about underfitting when $J_{train}(\theta)$ and $J_{cv}(\theta)$ are high, while we talk about overfitting when $J_{train}(\theta)$ is low and $J_{cv}(\theta)$ is quite higher than $J_{train}(\theta)$.

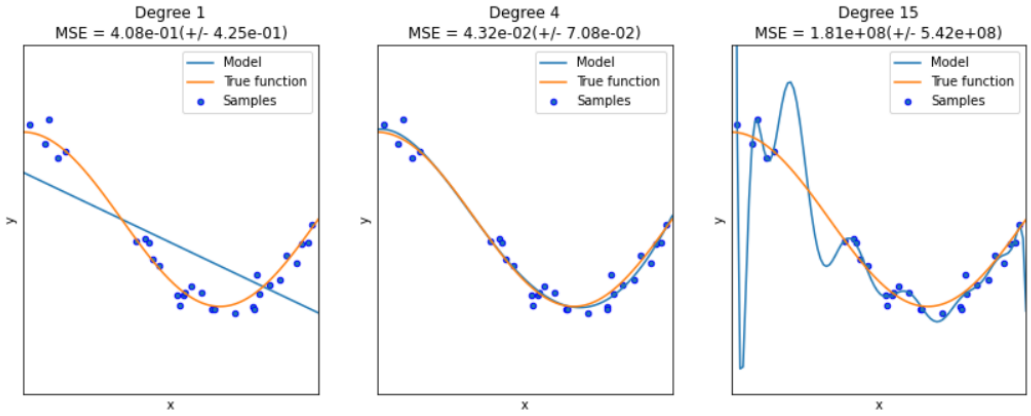


Figure 37:Underfitting and overfitting example [17]

The terms overfitting and underfitting therefore describe how much the model learns and generalizes the new data. A good machine learning model must be capable of generalizing new data well.

With the term underfitting we refer to a model that has a high cost function, i.e. with poor performance.

With the term overfitting we refer to when the model a network is too tied to the training data and does not generalize about new data, so the resulting test cost function is higher.

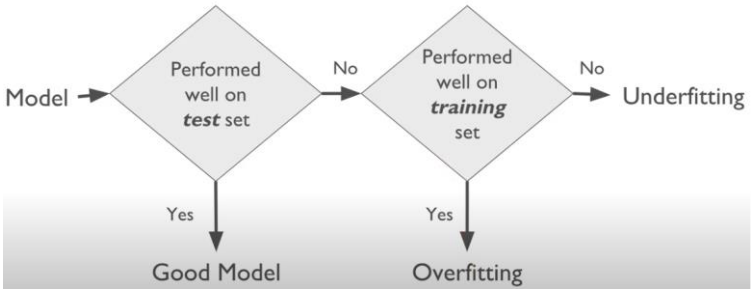


Figure 38: Logical scheme to understand if the neural network is affected by overfitting or underfitting

5. Creation And Optimization of the ANN

In this chapter we also talk about the optimization of the artificial neural network through the variation of its hyperparameters. The back-electromotive force coefficient signal k_{bemf} was obtained. About three thousand signals were analyzed and parameters representative of the health of the system were obtained. In the nominal operating condition (in which faults are not present) the value is unitary.

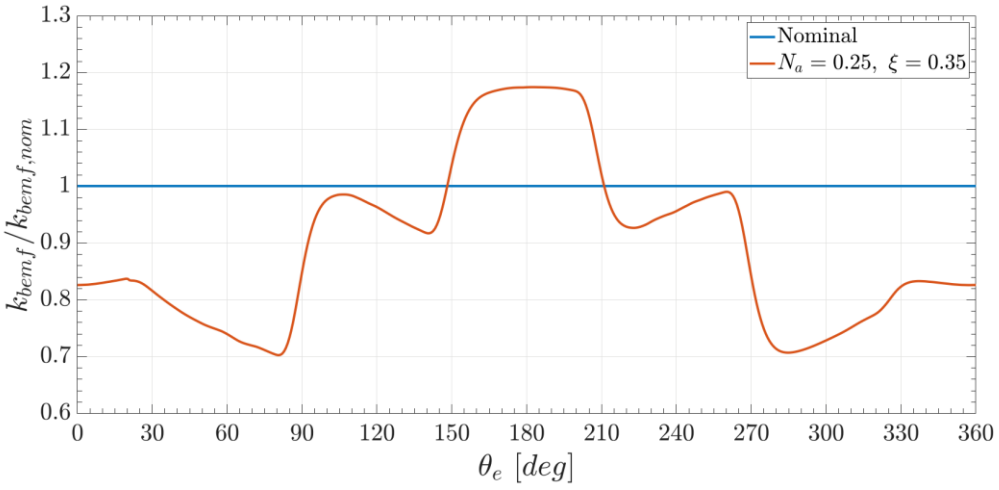


Figure 39: Back-EMF in nominal and faulty condition [18]

The signal has been appropriately sampled and each curve is represented by a certain number of points that characterize the features of the artificial neural network.

The latter must predict five output values:

$$[N_a, N_b, N_c, \xi, \phi]$$

1. N_a, N_b, N_c which represent the percentage coil short for each phase (the values are within the range [0,1] where 0 represents no damage and 1 total phase short).
2. $\xi = \frac{x_0}{g_0}$ which represents the static eccentricity, i.e. the relationship between the axis offset from center x_0 and the nominal air gap g_0.

3. ϕ which represents the angular phase of the static eccentricity.

For this reason, a single example of the training set is characterized by eighteen features and five outputs. [18]

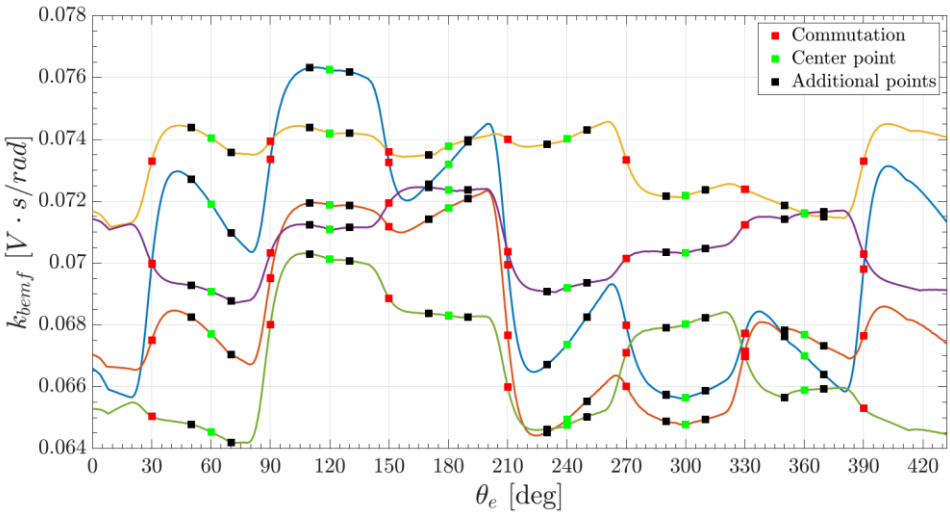


Figure 40: Sampling of five examples of back-EMF signals

At this point the network is trained using the Deep Learning Toolbox present in MATLAB from $MathWorks^{®}$.

Let us start now with a configuration suggested by [18].

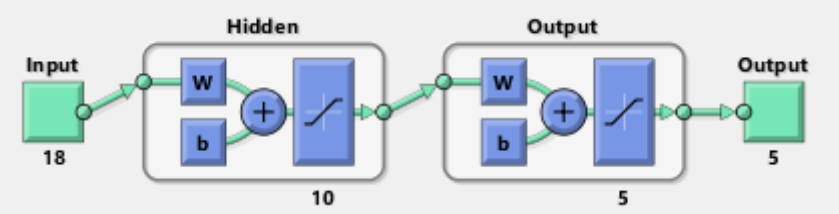


Figure 41: Architecture of the reference network

The network is made up of 3 layers:

- an input layer with 18 features representing the sampled signal.
- a hidden layer with 10 neurons.
- an output layer with 5 neurons representing $[N_a, N_b, N_c, \xi, \phi]$.

The training function used is '*trainlm*'. This function updates weight and bias values according to Levenberg-Marquardt optimization rule:

$$\boldsymbol{x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e}$$

Equation 5.1: Levenberg-Marquardt Algorithm update rule

where μ is a scalar value like the inverse of learning rate: a high μ value results in a gradient descent with a small step size and viceversa. The algorithm can manage this parameter automatically.

The Levenberg-Marquardt algorithm is designed to approach second-order training speed without having to compute the Hessian matrix. In fact, the latter is approximated as:

$$\boldsymbol{H = J^T J}$$

Equation 5.2: Hessian matrix approximation

while the gradient is calculated as:

$$\boldsymbol{g = J^T e}$$

Equation 5.3: Gradient's calculation

The '*satlins*' is used as an activation function between one layer and another.

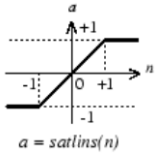


Figure 42: Satlins activation function

$$\boldsymbol{\phi(z)} = \begin{cases} \mathbf{1} & \boldsymbol{if\ z > 1} \\ \boldsymbol{z} & \boldsymbol{if\ -1 < z < 1} \\ \mathbf{-1} & \boldsymbol{if\ z < -1} \end{cases}$$

Equation 5.4: Satlins activation function

The results obtained are now presented:

- Simulation time: 2 s

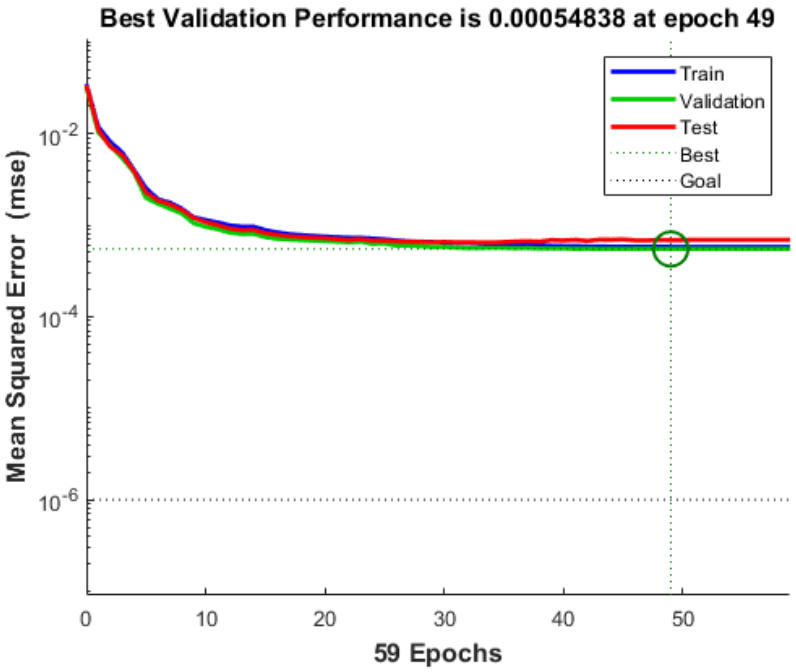


Figure 43: MSE performance of reference network

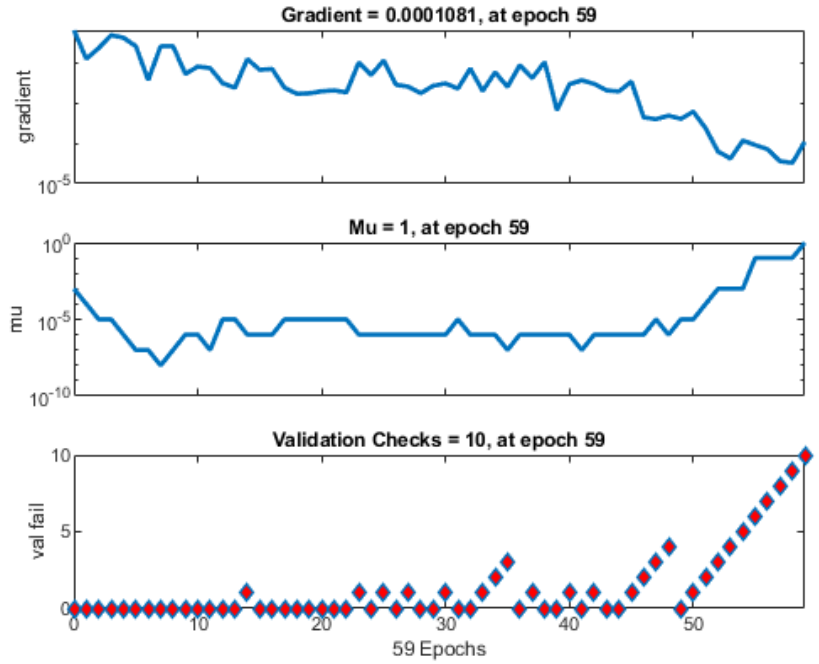


Figure 44: Results for reference network

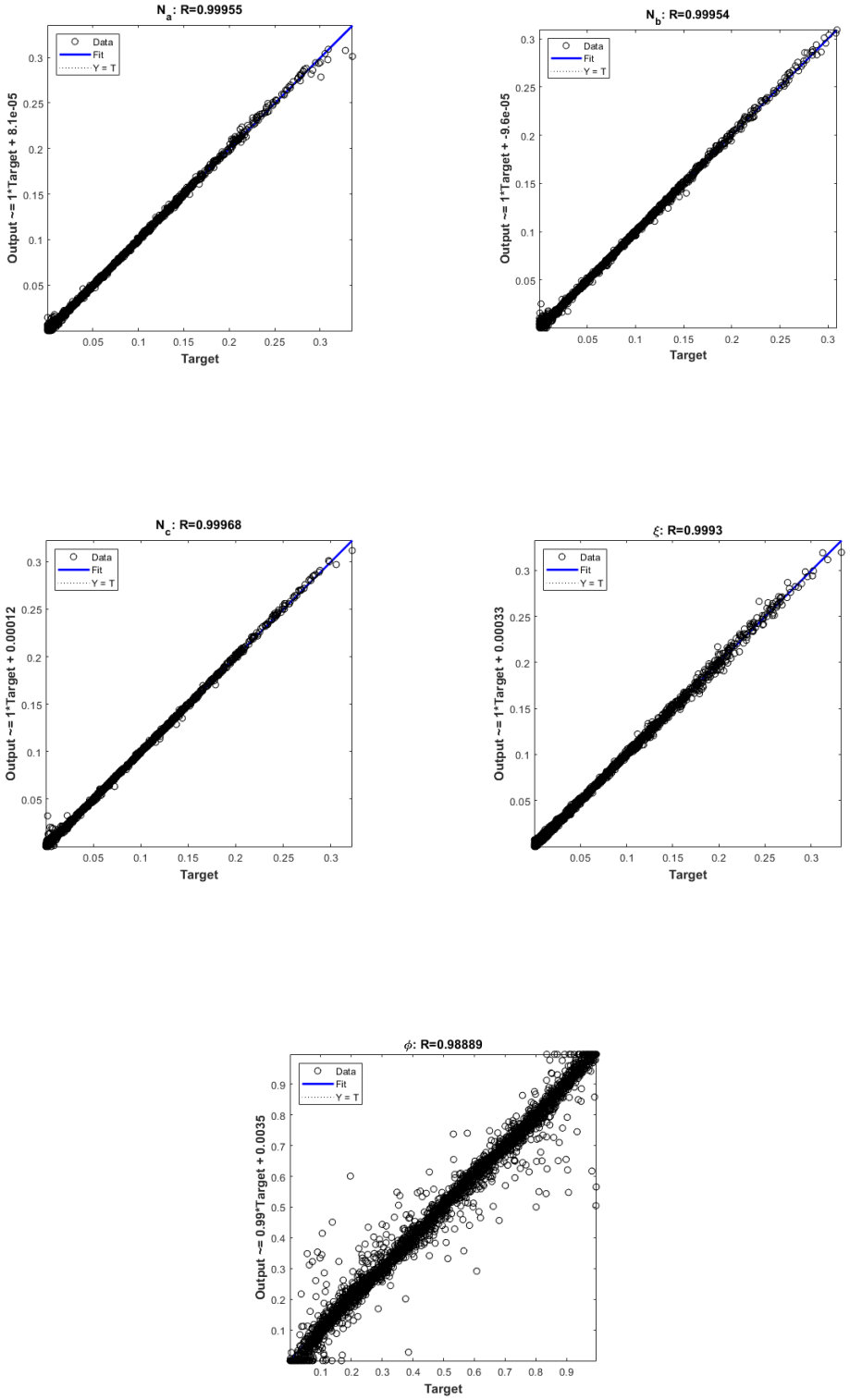


Figure 45: Linear regression plot for each output $[N_a, N_b, N_c, \xi, \phi]$

From the figures, we can conclude that the lowest value of the mean squared error (mse) for the validation cost function is reached at epoch 49 and is equal to 0.00054838.

The gradient never drops below the value of 10^{-5} and the simulation is stopped when 10 validation failures occur, i.e. the network does not improve or remains the same for 10 epochs.

Finally, from the graphs of the linear regressions we can see that the parameter that presents a greater error is the angular phase of the static eccentricity ϕ.

5.1. Variation of hyperparameters

Some hyperparameters of the network are now changed. In particular, the architecture is changed by varying the number of neurons and the number of layers.

Since we want to try to improve the model from the point of view of bias (underfitting), more complexity is added to the network. We then proceed with the variation of the activation function between one layer and another and finally with the introduction in the model of the regularization and the increase in the number of features. An increase in the complexity of the system leads to a rise in the computational time for training the network.

5.1.1. Neurons

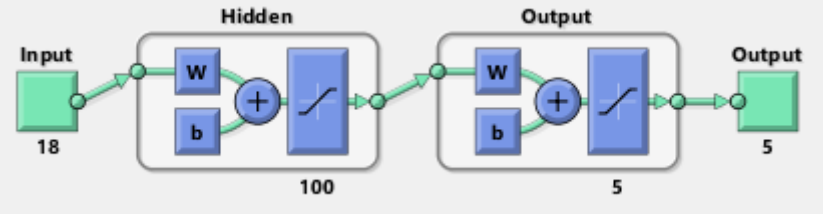


Figure 46: Increase in the number of neurons in the hidden layer

- Simulation time: 165 s

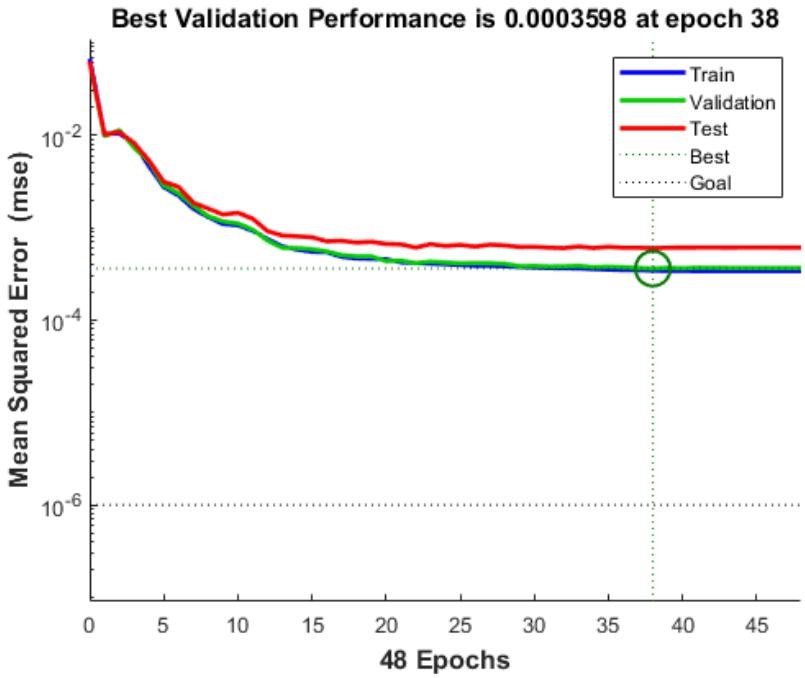


Figure 47: MSE performance for the network with with twenty-four neurons in the hidden layer

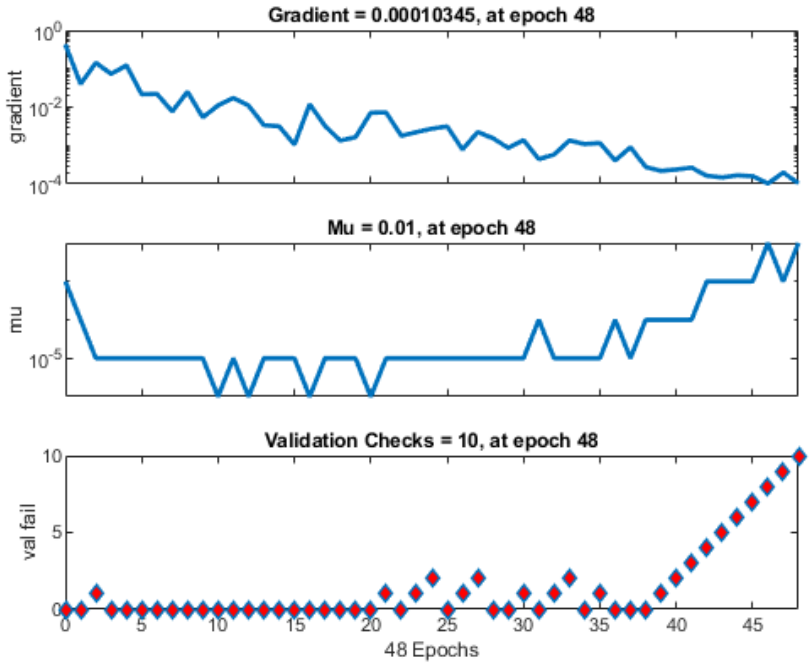


Figure 48: Results for the network with twenty-four neurons in the hidden layer

We can see improvements in network performance at the expense of more computational time.

5.1.2. Hidden Layers

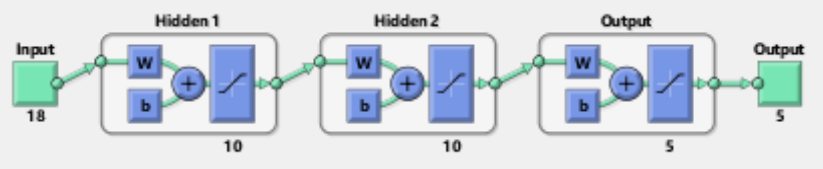


Figure 49: Increase in the number of hidden layers

- Simulation time: 3 s

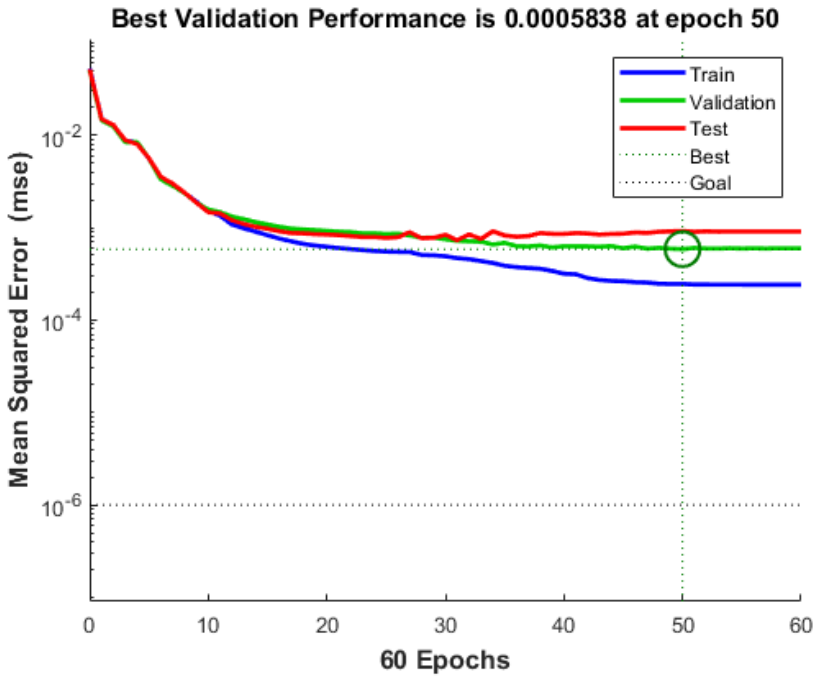


Figure 50: MSE performance for the network with two hidden layers

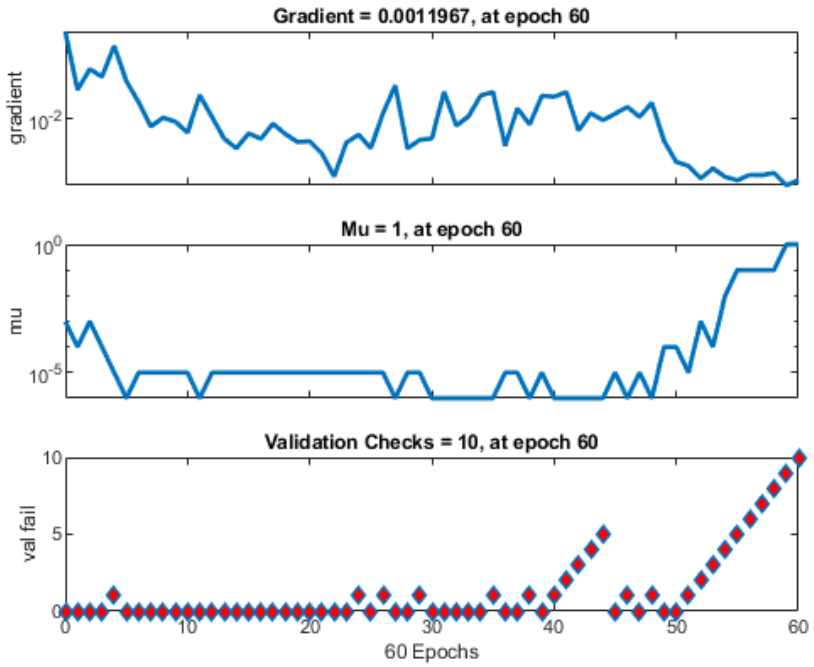


Figure 51: Results for the network with two hidden layers

5.1.3. Activation Function

Figure 52: Tansig activation function

$$\boldsymbol{\phi(z) = tanh(z) = \frac{sinh(x)}{cosh(x)} = \frac{e^{x}}{e^{-x}+e^{x}} - \frac{e^{-x}}{e^{-x}+e^{x}}}$$

Equation 5.5: Tansig activation function

In MATLAB we can use the 'tansig' function which corresponds to the tanh function as an activation function.

- ❖ Simulation time: 1 s

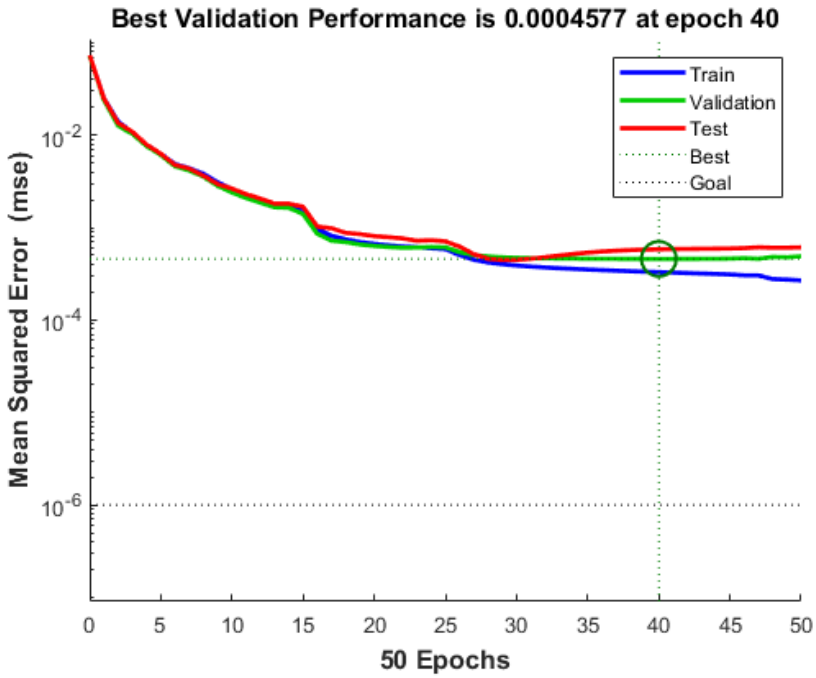


Figure 53: MSE performance for the network with tansig activation function

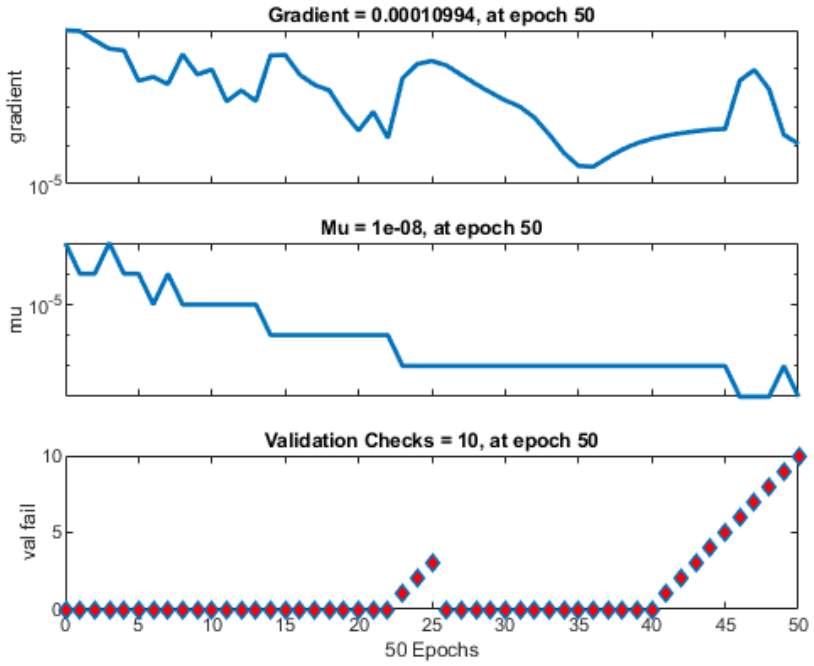


Figure 54: Results for the network with tansig activation function

5.1.4. Regularization

If the overfitting problem is caused by a neural network with many features, we can use regularization to solve this problem.

Regularization penalizes weights that become too high during training by means of coefficients. There are two types of regularizations: L1 and L2.

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

Equation 5.6: Cost Function with regularization L2 term

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}|\theta_j|$$

Equation 5.7: Cost Function with regularization L1 term

Usually the L2 regularization is more effective than the other and if we want, we can combine the two types of regularization.

The regularization L2 hyperparameter λ has been set to the value of $1 \cdot 10^{-5}$.

- Simulation time: 1 s

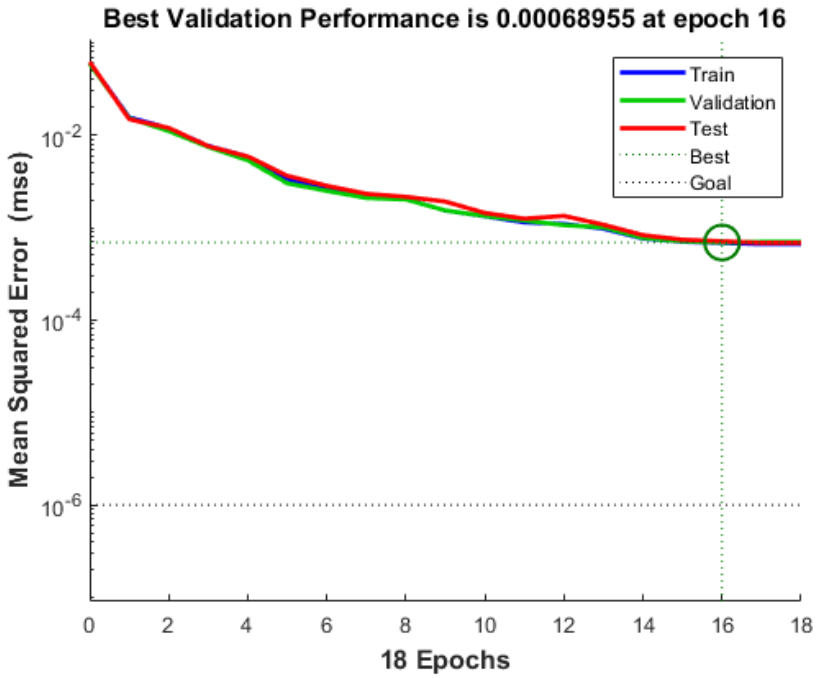


Figure 55: MSE performance for the network with regularization term

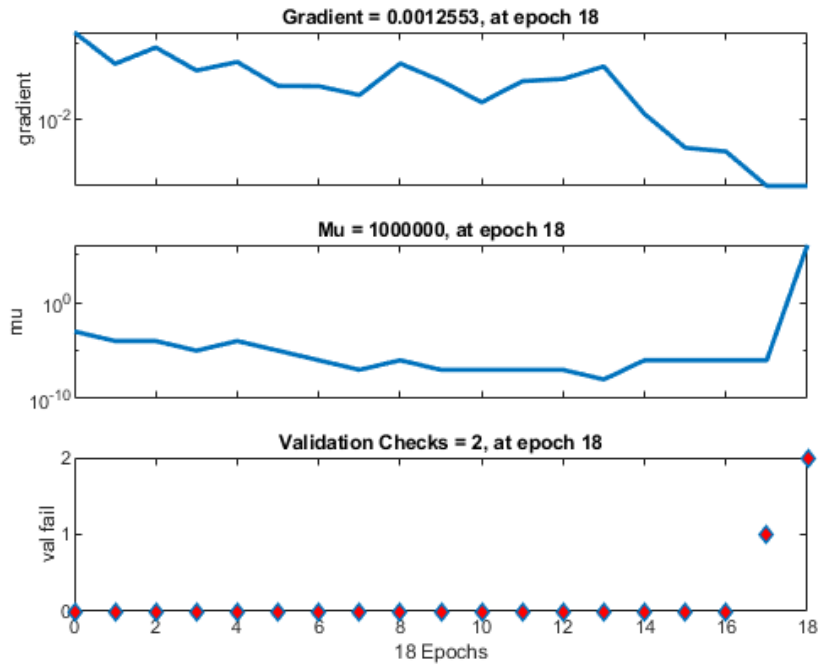


Figure 56: Results for the network with regularization term

In this case the simulation was interrupted by a μ value equal to 10^5.

5.1.5. Features

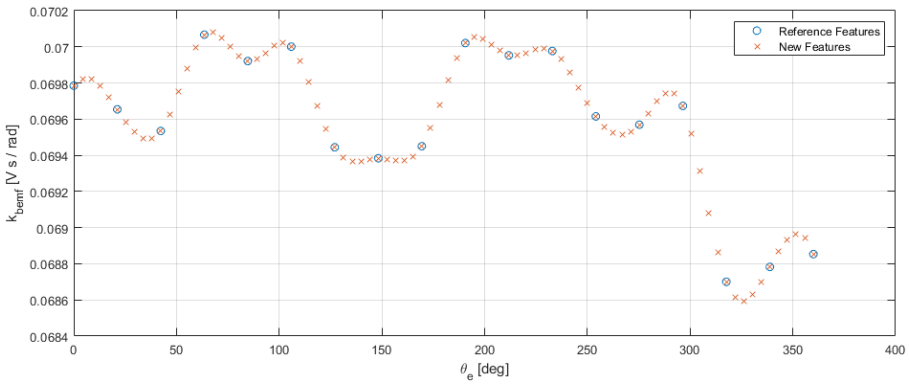


Figure 57: Back-EMF signal interpolation

From the reference signal of the back-EMF represented by eighteen features, thanks to the spline function, further points of the curve are obtained to have a greater number of features to train the neural network. We now have eighty-six features available.

- ❖ Simulation time: 27 s

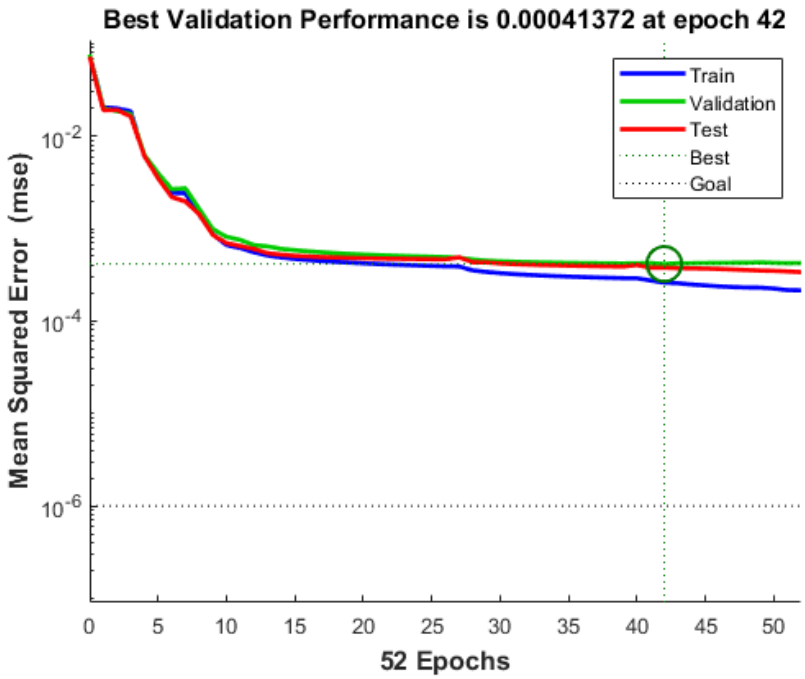


Figure 58: MSE performance for the network with new features

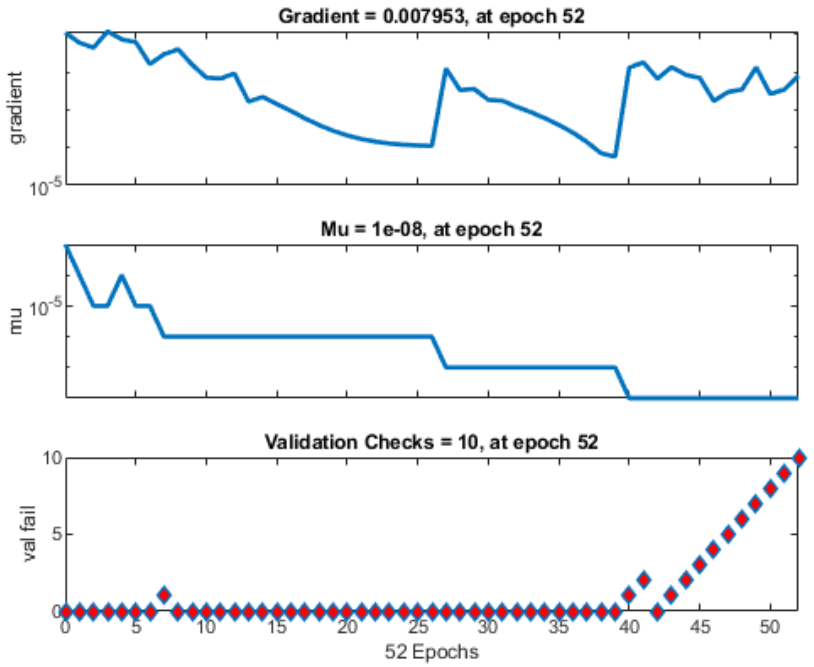


Figure 59: Results for the network with new features

5.2. Best Cases

Having seen how each individual hyperparameter affects the neural network, the best hyperparameter trade-offs are now presented.

5.2.1. Case 1

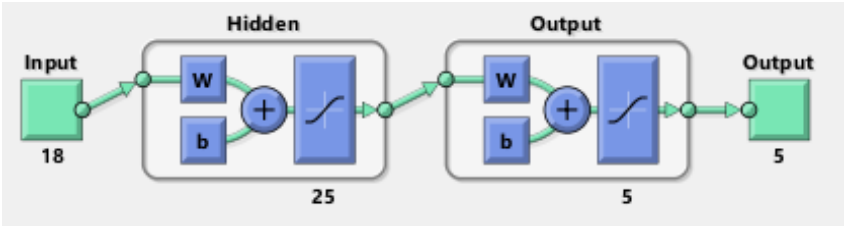


Figure 60: Network architecture for best case

- Simulation time: 11 s

net.layers{:}.transferFcn = 'tansig';

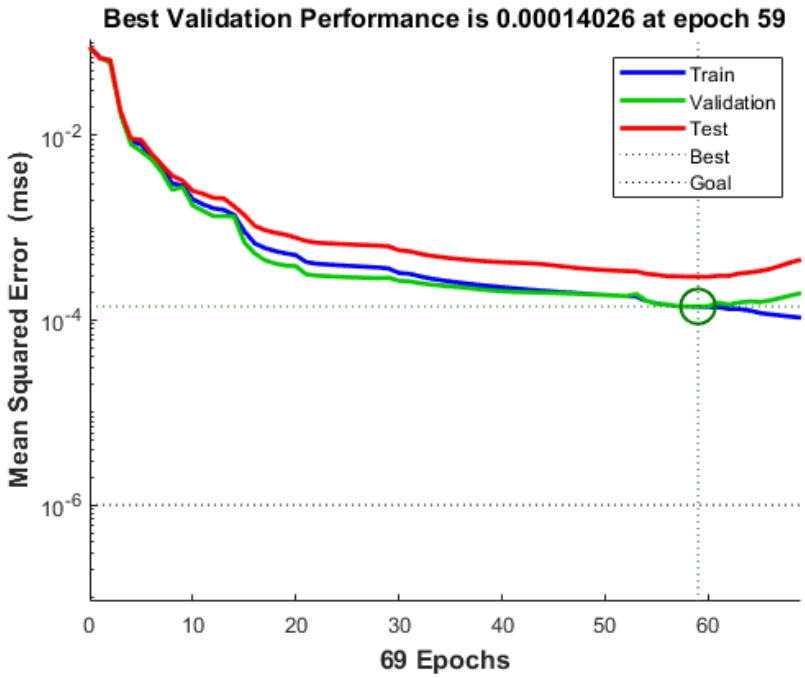


Figure 61: MSE performance for the best network

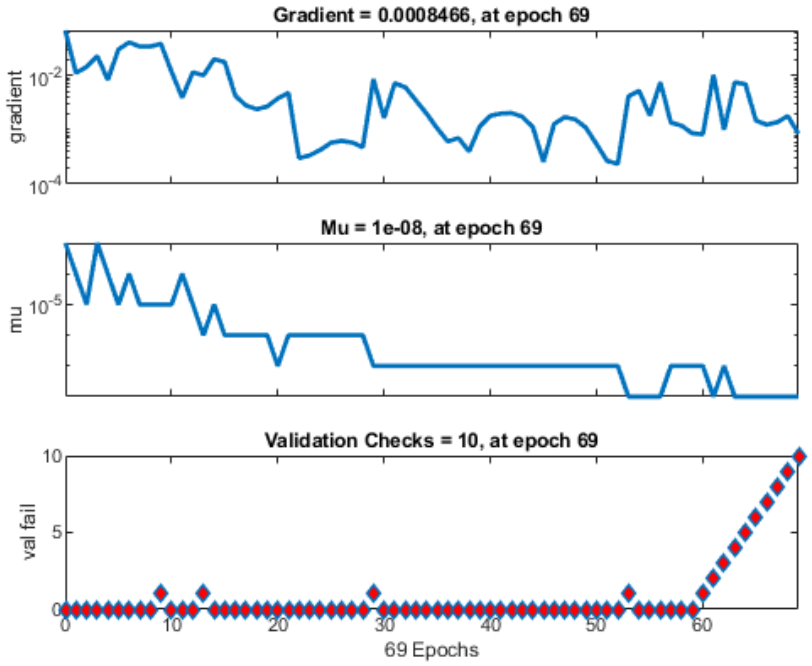


Figure 62: Results for the best network

5.2.2. Case 2

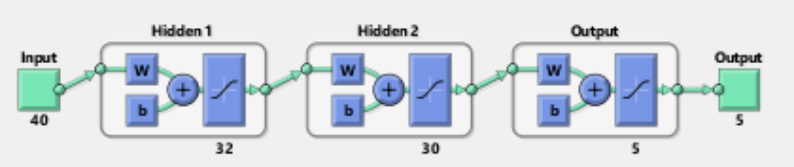
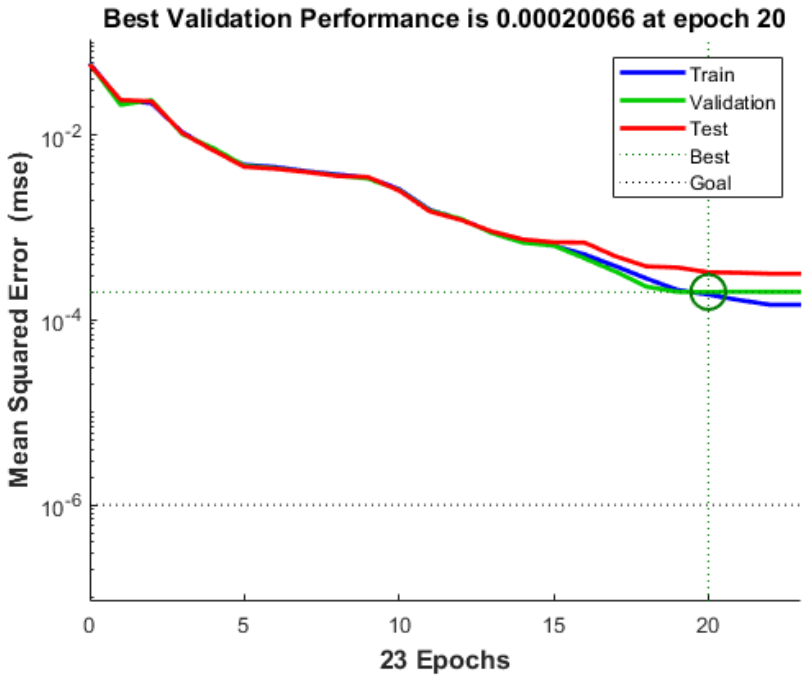


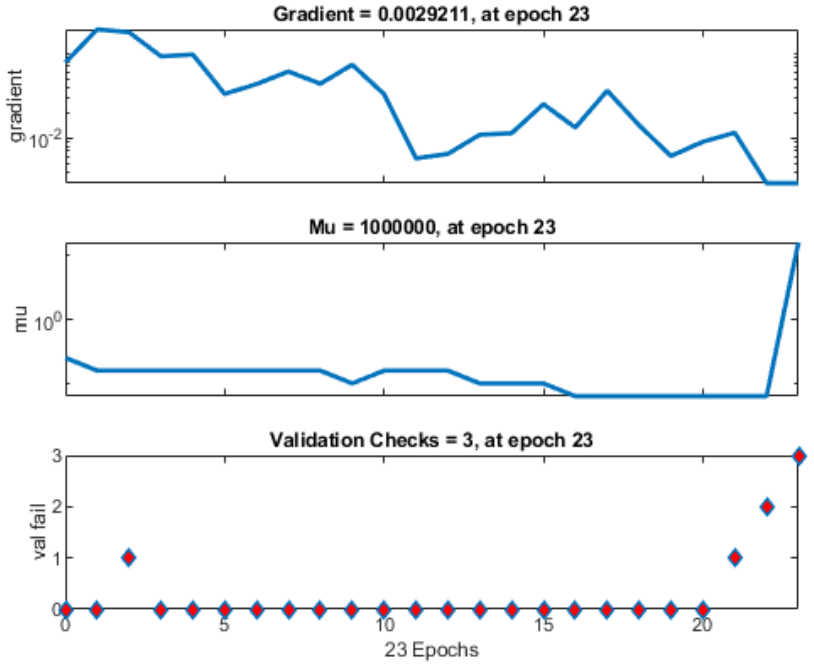
Figure 63: BEMF with 40 features

- Simulation time: 86 s

net.performParam.regularization = 5e-5;

net.layers{:}.transferFcn = 'tansig';




Gradient = 0.0029211, at epoch 23
gradient
10^{-2}
Mu = 1000000, at epoch 23
mu
10^{0}
Validation Checks = 3, at epoch 23
val fail
0
1
2
3
0
5
10
15
20
23 Epochs

CONCLUSIONS

INDEX OF FIGURES

INDEX OF TABLES

INDEX OF EQUATIONS

BIBLIOGRAPHY

[1] M. D. Dalla Vedova, A. Germanà, P. C. Berri e P. Maggiore, «Model-Based Fault Detection and Identification for Prognostics of Electromechanical Actuators Using Genetic Algorithms,» *MDPI,* 2019.

[2] G. Quattrocchi, Development of innovative prognostic methods for EMAs, 2019.

[3] F. Dabbene, *Introduction to Dynamic Systems Modeling,* 2019.

[4] Dejan, «How Brushless Motor and ESC Work,» How to Mechatronics, [Online]. Available: https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/.

[5] B. L. Stevens, F. L. Lewis e E. N. Johnson, Aircraft Control and Simulation Third Edition, Wiley, 2016.

[6] C. N. D. a. D. P.M. Churn, «Elkctro-hydraulic actuation of primary fliyht control surfaces,» *IEE Colloquium on All Electric Aircraft,* 1998.

[7] R. C. Munjulury, «Model Based Aircraft Control System Design and Simulation,» 2009. [Online]. Available: https://www.researchgate.net/publication/268241518_Model_Based_Aircraft_Control_System_Design_and_Simulation.

[8] S. Corpino, *Corse notes of "Gestione dei Rischi, Costi e Supporto Logistico Integrato dei Sistemi Aerospaziali",* 2019-2020.

[9] D. Z. J. W. Q. J. W. Tao He, «Experimental and Numerical Investigations of the Stribeck Curves for Lubricated Counterformal Contacts,» *Journal of Tribology,* 2016.

[10] M. D. V. L Borello, «Dry friction discontinuous computational algorithms,» *International Journal of Engineering and InnovativeTechnology (IJEIT)3.8,* 2014.

[11] I. The MathWorks, «Matlab Wavelet Toolbox,» [Online]. Available: http://matlab.izmiran.ru/help/toolbox/wavelet/ch06_a47.html.

[12] S. V. Vaseghi, Advanced Digital Signal Processing and Noise Reduction, 2006.

[13] M. Battipede, M. D. Dalla Vedova, P. Maggiore e S. Romeo, «Model based analysis of precursors of electromechanical servomechanisms failures using an artificial neural network,» *IAA Modeling and SimulationTechnologies Conference,* 2015.

[14] M. Nielsen, Neural Networks and Deep Learning, 2019.

[15] P. A. Giuseppe Gullo, «Deep Learning e Reti Neurali con Python: il Corso Pratico,» 5 2020. [Online]. Available: https://www.udemy.com/course/deep-learning-pratico/.

[16] A. Ng, «Machine Learning,» Stanford University with Coursera, 2020. [Online]. Available: https://www.coursera.org/learn/machine-learning.

[17] M. Tripathi, «Underfitting and Overfitting in Machine Learning,» Datascience Foundation, 2020. [Online]. Available: https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning.

[18] G. Quattrocchi, P. C. Berri, M. D. Dalla Vedova e P. Maggiore, «Innovative Actuator Fault Identification Based on Back Electromotive Force Reconstruction,» *mdpi,* 05 2020.

[19] P. Maggiore e M. D. Dalla Vedova, *Appunti del Corso di "Modellazione, simulazione e sperimentazione dei sistemi aerospaziali",* 2019-2020.

[20] D. Wilson, «So, Which PWM Technique is Best,» Texas Instruments, 2012. [Online]. Available: https://e2e.ti.com/blogs_/b/industrial_strength/archive/2012/04/13/so-which-pwm-technique-is-best-part-6.