POLITECNICO DI TORINO

Corso di Laurea Magistrale

in Ingegneria Aerospaziale

Tesi di Laurea Magistrale

Dragonflies Algorithm integration

as a prognostic method for EMA

Relatori

prof. Matteo Davide Lorenzo Dalla Vedova

prof. Paolo Maggiore

Candidato

Carlo Dimby Mosa

Anno accademico 2019/2020

Abstract

The more-electric philosophy for aerospace technologies brought interest in the application of electronics actuation systems which are able to replace the old hydraulic servomechanism. Such conversion comes with different problems alongside, in fact it is absolutely mandatory the study of the possible faults mode in order to preserve the reliability and safety of the overall aircraft system. To do so, a model-based approach for prognostic applied to the electromechanical servosystem, responsible for the movement of a generic flight control surface, has been studied in my university and various works focused on the first phase of this process, called fault detection & identification, were produced.

The EMA actuator presented is modeled with an high-fidelity and nonlinear model called reference model (RM) which is capable of represent the physics of the system accurately and provide a cheap and fast way to analyze the function of this actuator, but the computation cost is very heavy.

In order to study the remaining useful life and eventually proceed with corrective action, a lighter and linear model called monitor model (MM) was developed, it requires an optimization to achieve the capability of simulate the faulty RM accurately. In this work the Dragonfly Optimization is applied to the monitor and tested.

Thanks

Vorrei ringraziare innanzitutto l'ingegnere Dalla Vedova e il prof. Maggiore per avermi dato l'oppurtinità di svolgere la tesi su un arogmento così interessante. Nonostante le difficoltà affrontate, ho avuto modo di conoscere e di approfondire una disciplina a me già cara da cui ho imparato molto, e per questo vi sono grato.

Ringrazio inoltre la mia famiglia, sempre presente al mio fianco.

Colgo l'occasione per salutare e ringraziare I miei cari amici e colleghi Gyca, Nicholas e Davide, che durante il tragitto mi hanno supportato sempre.

Contents

Bibliography

- S.Mirjalili, “Dragonfly algorithm: a new metha-heuristic optimization technique for solving single-objective,discrete,and multi-objective problems”, The Natural Computing Application Forum, 2015.
- S.Re, “Development and comparison of prognostic methodologies applied to electromechanical servosystems (EMA) for aerospace purpouse”, DIMEAS, Politecnico di Torino, 2018.
- P.C. Berri, “Genetic algorithms for prognostics of electromechanical actuators”, DIMEAS, Politecnico di Torino, 2016.
- P.C. Berri, M.D.L Dalla Vedova, P. Maggiore, “On-board electromechanical servomechanism affected by progressive faults: proposal of a smart GA model-based prognostic method”, DIMEAS, Politecnioc di Torino.
- M.D.L. Dalla Vedova,P. Maggiore, L. Pace, S. Romeo, “ Proposal of a model based fault identification neural technique for more-electric aircraft flight control EMA actuators”, DIMEAS, Politecnico di Torino, 2016.
- T.Sutharssan, S. Stoyanov, C. Bailey, C. Yin, “ Prognostic and health management for engeeniring systems: a review of data-driven approach and algorithms”, The Journal of Engineering, 2015

1.Introduction

1.1 An introduction to prognostics

The aerospace industry has always demanded high levels of reliability and safety for its products and proper techniques to meet such requirement in short times with low cost, the actuation system, that will be presented after, it's used to move a secondary control surface hence it represent one critical component of the aircraft which durability must be conserved by avoiding critical failures.

The Prognostics and health management is the discipline that links studies of failure mechanisms to system actual life cycle. This work is focused only on the prognostics, used to predict the remaining useful life (RUL) of a system (the amount of time left to the system in which it can operate in nominal condition) and proceed with cost and maintenance planning.

The prognostics follow the next steps:

- DATA COLLECTION: the condition of the system is monitored with sensors that collect useful data, in this case the dynamic response of the actuators is described by position, speed, acceleration and currents. Failures of the detection capability of the sensors must be prevented to avoid contamination of the database with error affected variables.
- TRENDS CLEAN-UP: while operating, sensors measurement can be affected by different factors like noise or vibration, resulting in an inaccurate output generation that doesn't fit with the real behaviour of the system. To avoid this, filters are installed to clean up the signal containing the detected variable before it is collected.
- THRESHOLD: The value of the data must fit in a precise interval, this one determined by requirement, specification or history.
- PREDICTION OF THE RUL: the most important phase, this can be achieved with different techniques combining experience, mathematical expressions, statics and more to estimate the future behaviour of the systems and enhance its reliability by planning maintenance.

Two different type of prognostics exist: Data-driven and Model-based.

In this thesis a model-based approach is presented, the RF model is well suited to describe the real behaviour of the systems but is huge complexity prevent multiple operation of data collections, in particular it is hard to perform multiple fault detection and so a monitor model, faster and simplified, can be optimized with algorithms to achieve similar performance and speed-up the whole prognostic process.

1.2 Flight controls

The Flight controls are divided in primary and secondary, they are regulated by the pilot and used to control the aircraft. This is possible thanks to the aerodynamic forces and torques obtain by the movement of the primary control surfaces while the secondary serves are responsible for the attitude control and other functions.

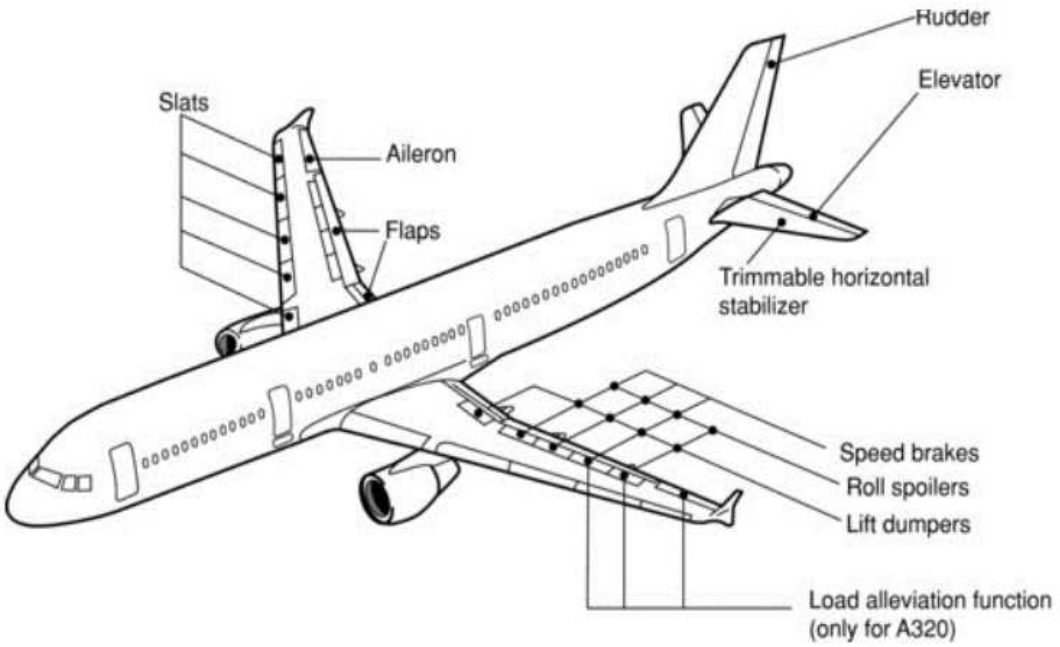


Fig 1.1: Primary and secondary flight controls

1.2.1 Primary Flight controls

These are the most critical controls, for modern civil aircraft they are heavily controlled by a dedicated system to achieve better performance than the pilot, even when the pilot is actually in handing the aircraft the Flight control system is still active to provide aerodynamic compensation and stabilization of the dynamic response of the surfaces. The controls are slit on 3 axis in this way:

- Pitch control, performed by the parallel actuation of the elevators situated on the tail.
- Roll control, obtained by the antagonist movement of the ailerons, situated on the inboard wing section.
- Yawn control, performed by the rudder situated on the tail.

Other kind of surfaces exist and architectures exists but are not presented here.

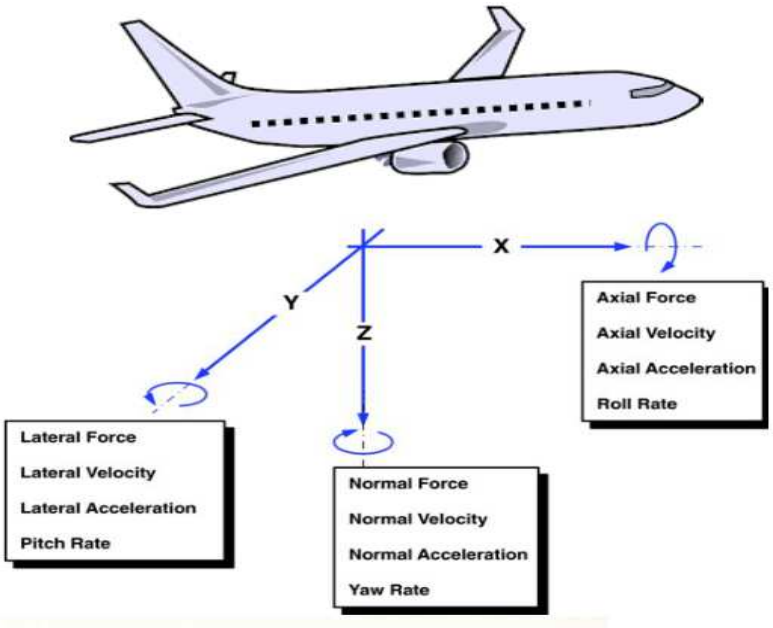


Fig 1.2 Forces and rotations

1.2.2 Secondary flight controls

These are less important and also less critical than primary controls ,since they are not used to directly control the aircraft, the secondary flight controls are mainly used for the regulation of aerodynamics attitude during the various phases of the flight and can on or off, there are the elements:

- Slat: placed overboard, it is used to achieve a greater maximum coefficient of lift, so the wing can avoid the stall.
- Flap: the most common, it is placed in the inboard wing section and has the purpose of increasing the lift while is active.
- Spoiler and airbrake: placed ahead of flaps, their function is to increase drag.

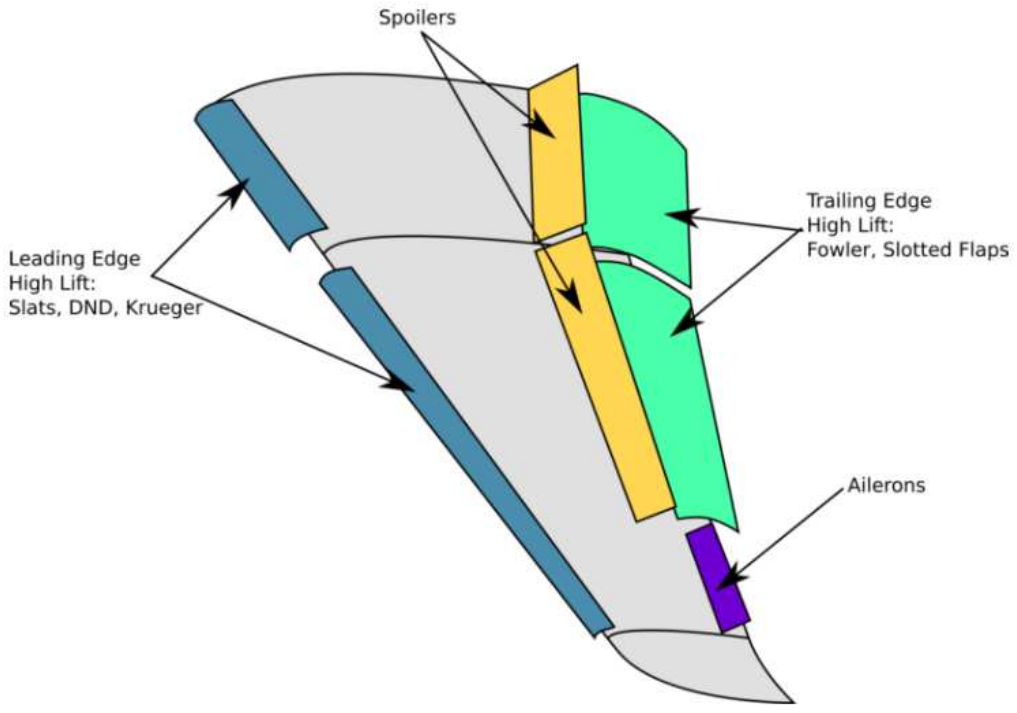


Fig 1.3 Secondary control surfaces

1.3 Actuation systems for control surfaces

The servomechanisms utilized an input signal, which natures depend on the actuation system, that moves the controlled surface to the required position. The input is generated and confronted to the actual position of the surface that is feed-backed,so every actuation system behaves as a closed-loop system. These feature is mandatory and add an high level of reliability to the flight controls, further increased by the presences of redundancies.

1.3.1 Hydromechanical actuation system

The Hydro-mechanical actuation represents the first solution adopted to control the flight controls. The input is given by the pilot and elaborated by a servo-valve which purpose is to adjust the hydraulic pressure given by the oil in the actuator chambers, then the pressure differential obtained moves the piston, which is connected by a hinge to the control surface in order to achieve its rotation.

This servomechanism is classified by the controlled physical dimension, it can be:

- Position (linear or angular)
- Speed (linear or angular)
- Force (also momentum or pressure)

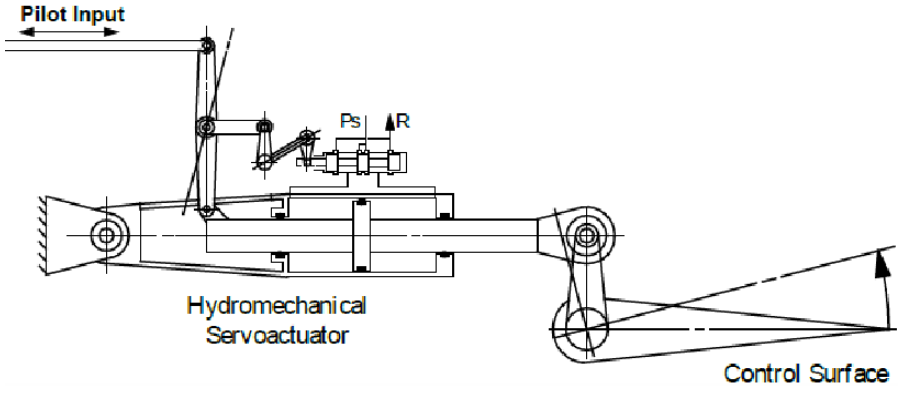


Fig. 1.4 hydromechanical actuator

1.3.2 Electrohydraulic actuation system

This represent a modern and deeply used technology, the input is electric, carried by a fly-by-wire structure on the servo-valve that can be one of the following types:

1) Direct drive valve

The electrical signal is commuted in a translation by a solenoid and a spool: this last element have the control the level of oil served to the user with the opening of tiny lights . The power level of the input signal must be high enough to move the mechanical part.

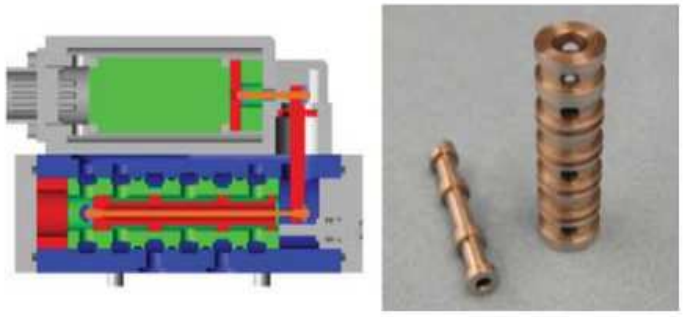


Fig.1.5 Direct drive valve

2) Flapper-nozzle

This valve require a low power input signal, its meaning is to move exert a little magnetic force on the flapper, the movement from is normal position induces an asymmetry on the oil in the valve chamber and this difference will move the spool. This kind of valve act like an amplifier of signal in fact.

Feedback is achieved by a pin to the end of the flapper that redirect the flapper to its natural position when the required position is meted.

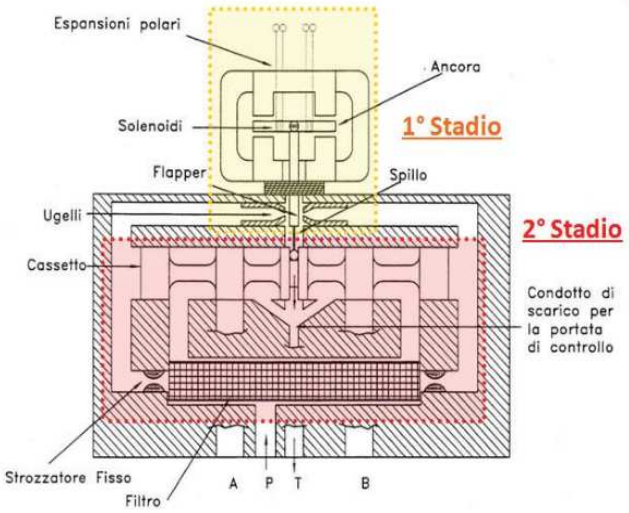


Fig 1.6: Flapper-nozzle valve

3) Jet-pipe

It works by the same principle of the flapper-nozzle but is regulated by an anchor that send the fluid to the spool, the feedback comes from a feedback spring that doesn't affect the spool but only the anchor. Without this, the valve can work only in on and off mode.

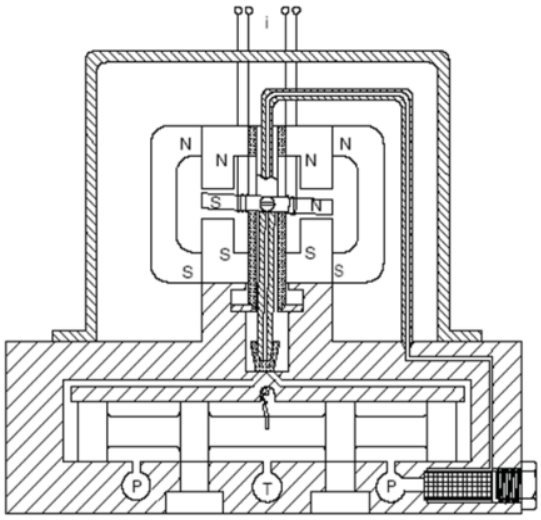


Fig. 1.7 Jet-pipe valve

1.3.3 Electrohydrostatic actuation system

Another viable solution is the EHA. It is a valuable choice for its independence from the hydraulic system, in fact, this is connected only to the electric system and has better performance in terms of reliability compared to the various hydraulic acutiation technologies.

The surface is moved with an hydraulic piston, this is controlled by a pump that receive power from a BLDC motor. The pressurize fluid is stored in a tank and controlled in pressure by dedicated valves, this circuit is therefore independent by the hydraulic system.

There are 3 solutions for this type of actuator:

- Fixed pump displacement and variable motor speed
- Variable pump displacement and fixed motor speed
- Variable pump displacement and variable motor speed

The fly-by-wire impact slightly on the weight of the aircraft and is not suited for military application due to the low frequency of it.

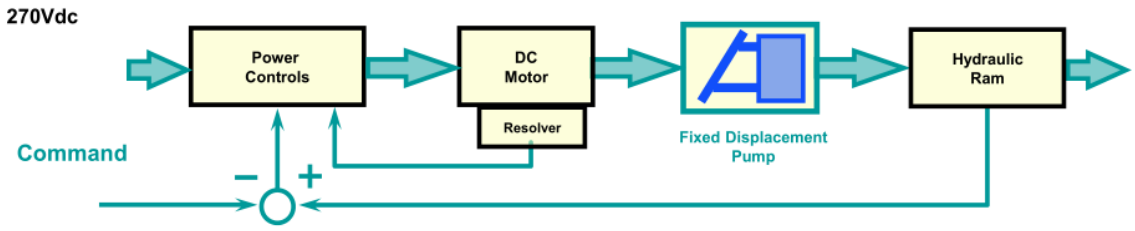


Fig. 1.8 control loop for EHA

Fig. 1.9 Electrohydraulic actuator

1.3.4 Electromechanical actuation system

The electromechanical solution is the actuation system studied in this paper, it works by commuting electrical power with the BLDC motor, this motor is linked to a reducer to adjust the rpm in favor of an higher torque output that will affect the ball screw, which converts the motion form rotational to linear and so enable the movements of the surface.

This solution is new and its faulty behavor is hard to simulate in a fast and effective way, therefore it is only used for secondary flight controls.

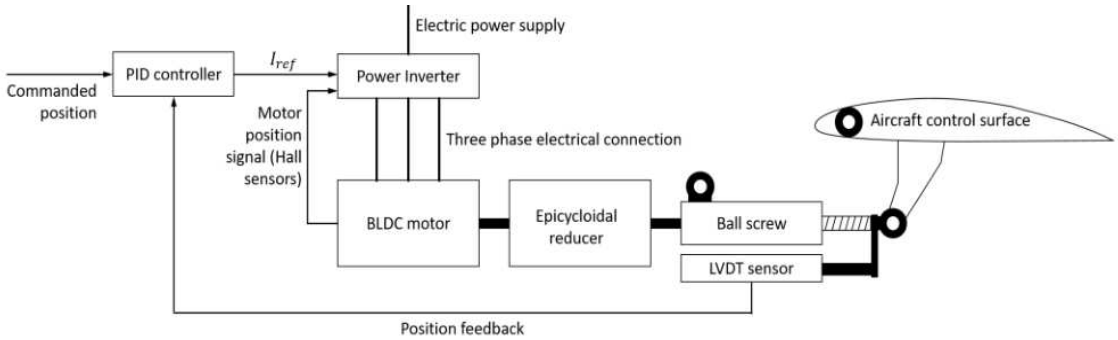


Fig. 1.10 Scheme of a EMA actuator

2.BLDC Motor

The brushless motor is a modern solution for electric-powered actuation systems.

This technology has replaced the old electric motor, which was provided with brushes paired with a collector, this kind of solution is able to provides current in the conductors of the rotor with the friction applied between brushes and coils. Such friction causes a fast deterioration of the components alongside noise and heat production, with another issue dependent of the production of conductive particles that may generate electric arcs.

The BLDC provides a cheaper, both for production and maintenance needs, and is also a more reliable solution.

Fig. 2.1 BLDC motor

2.1 Components

The BLDC motor is composed by a rotor and a stator, there are different solutions for the position of those based on the application, in this work is considered an outer rotor type were the magnets are positioned on the rotor while the winding are placed on the stator.

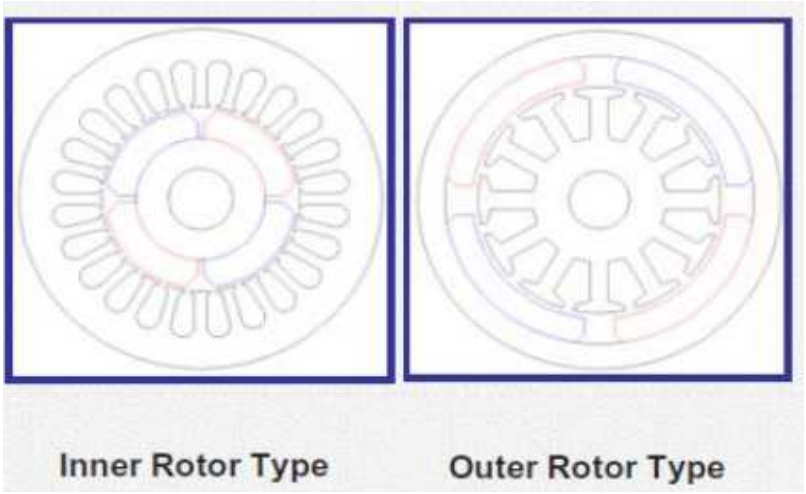


Fig. 2.2 type of BLDC motors

The number of winding defines the phases, for a three-phases motor the configuration is chosen between the star (Y) and the delta, the first provides a voltage equal to the line while the second has it (inserisci V diviso 3). The Y pattern is chosen for the studied application since the windings can be powered individually while the others are shut down.

Fig 2.3 Star and Delta configuartion

The number of magnets defines the pair-pole, increasing such number lead to better performances, with the same currents its torque will be stronger, and also higher cost. The material used for the magnets is also important in term of performances and is often an expensive rare earth alloy, for example the Samarium Cobalt.

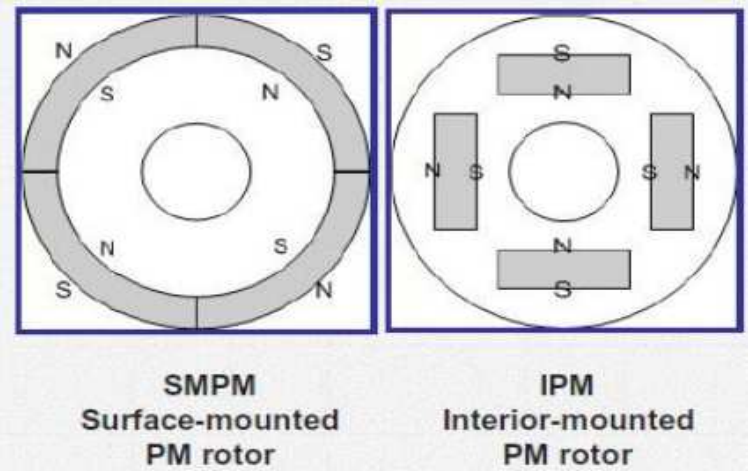


Fig.2.4 different magnet positioning in BLDC motors

2.2 Working principle

The motor is driven by a torque generated from the repulsive forces between magnets, this is achieved with the sequential powering of each winding that generate a magnetic field by interacting with the magnets on the rotor.

The power applied on a three phase motor is positive on the powered winding , while one is shut down and neutral, and the last one is negative (because the current is leaving). The whole process

is regulated and controlled, to solve this issues Hall sensors are mounted on the motors and provides the correct position of the rotor, thanks to the Hall law which state that a conductor that is pervaded by currents and immersed in a magnetic fields applies a traverse force to the passing charges, generating a tension on the conductor.

Three Hall sensors are placed on the stator with a 120° displacement.

The Hall tension provides a positive or negative currents depending on the state of the passing phase and so it is possible to locate the position of the rotor.

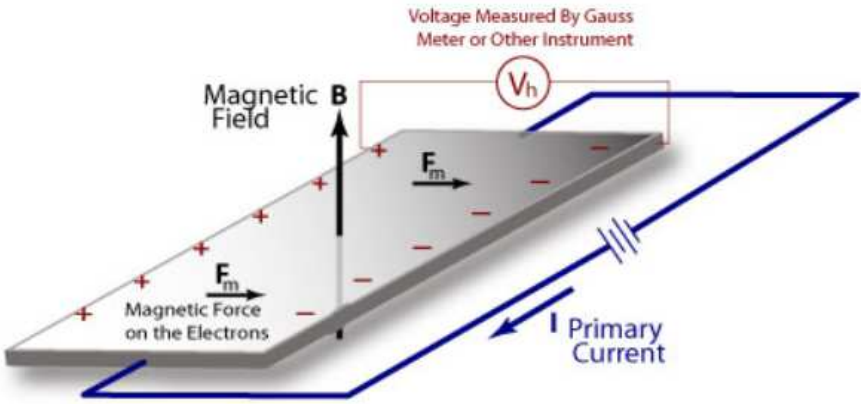


Fig. 2.6 Hall effect

Here is a scheme that shows how the phases moves the rotor;

Electrical position		$0^o < \Theta_e < 60^o$	$60^o < \Theta_e < 120^o$	$120^o < \Theta_e < 180^o$	$180^o < \Theta_e < 240^o$	$240^o < \Theta_e < 300^o$	$300^o < \Theta_e < 360^o$
Hall sensors	H1	1	1	1	0	0	0
	H2	0	0	1	1	1	0
	H3	1	0	0	0	1	1
Motor phases	A	off	ground	ground	off	supply	supply
	B	supply	supply	off	ground	ground	off
	C	ground	off	supply	supply	off	ground

Tab. 2.1 Switching sequencing

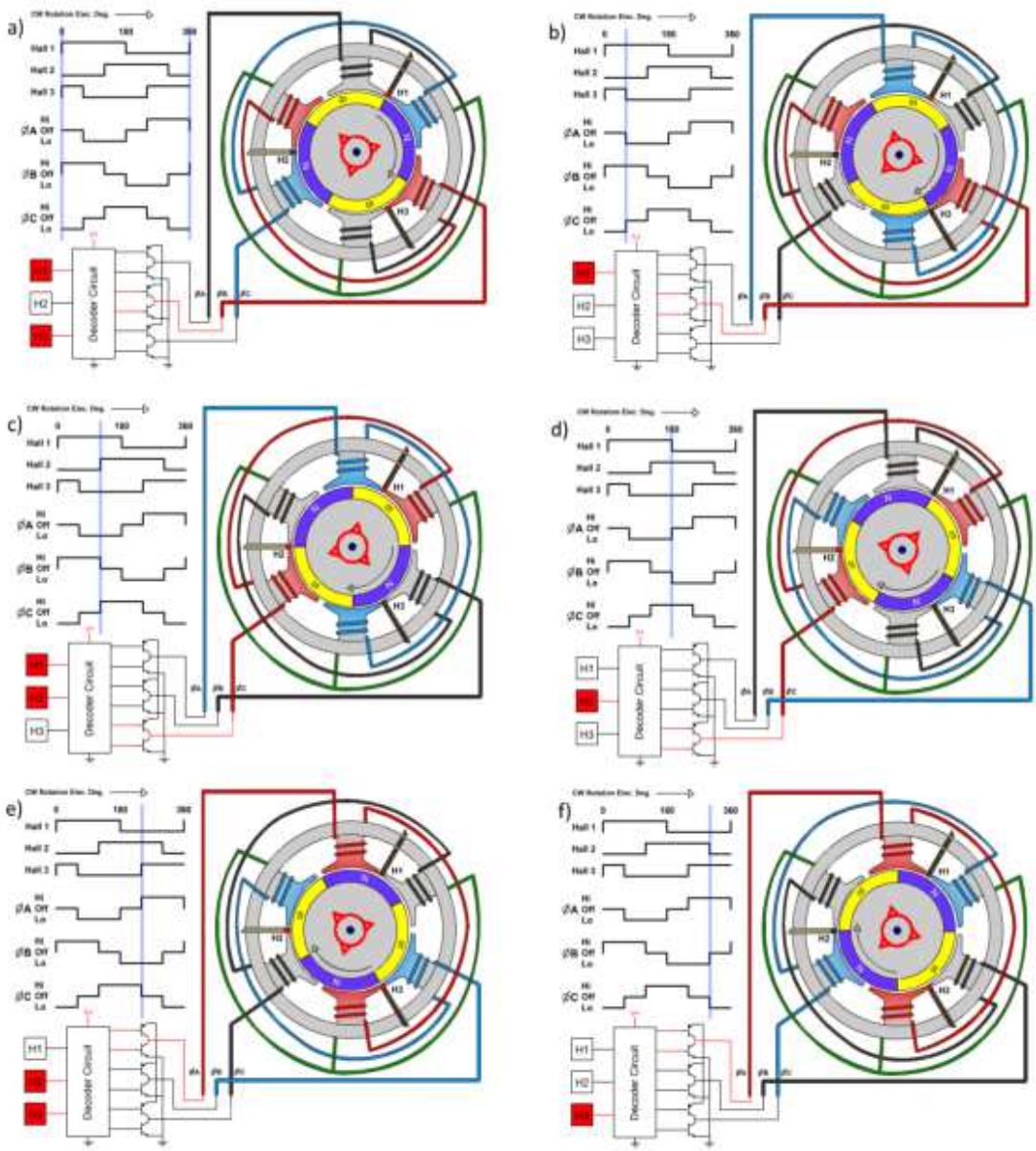


Fig. 2.7 Commutation sequencing in a BLDC motor

2.3 BLDC efficiency and output

The performances of the BLDC motor can be easily viewed on the characteristic graphics that shows how they are linked:

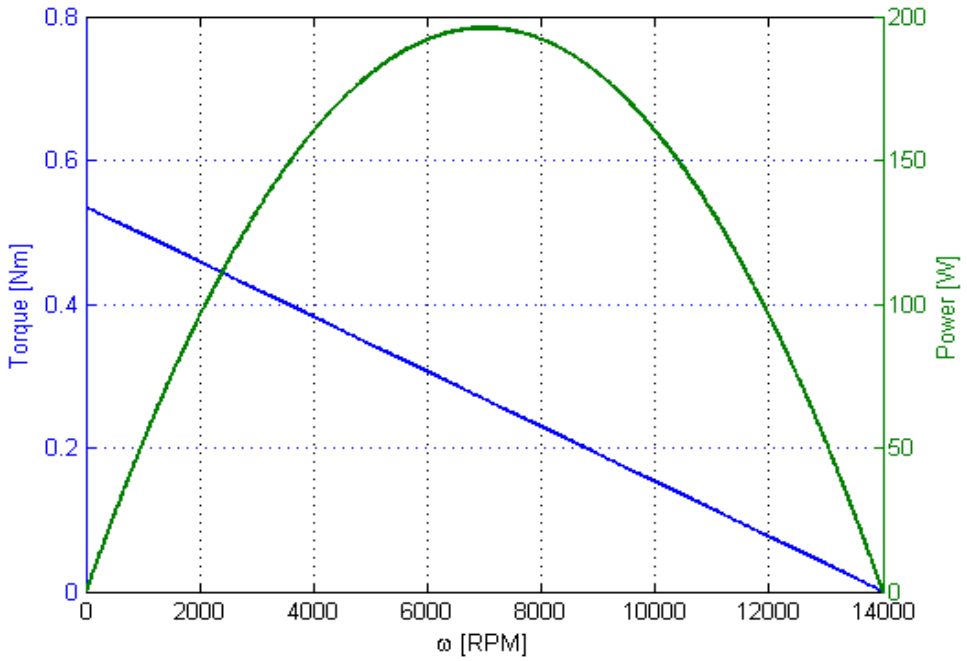


Fig. 2.8 Torque and power as function of the angular speed

The following relation shows how the torque depends on angular speed:

$$C = \frac{V \cdot Kc}{R} + \frac{Kc^2}{R}\omega$$

where Kc is the torque electric constant.

The power output is:

$$P = C \cdot \omega = \frac{V \cdot Kc}{R} + \frac{Kc^2}{R}\omega^2$$

The torque is linear and decreases with the angular speed, while the power has a quadratic dependence.

It is now clear that the highest torque is given by the motor when the angular speed is 0, also the maximum speed is reached when the torque is equal to 0.

The power input is P_i=Vi , so the efficiency is calculated as:

$$E = \frac{P_{out}}{P_i} = \frac{k_c \cdot \omega}{V}$$

In reality, both torque and speed are limited by physical issues such as heat degradation and mechanical limits of the rotor, the final characteristic graphics are below:

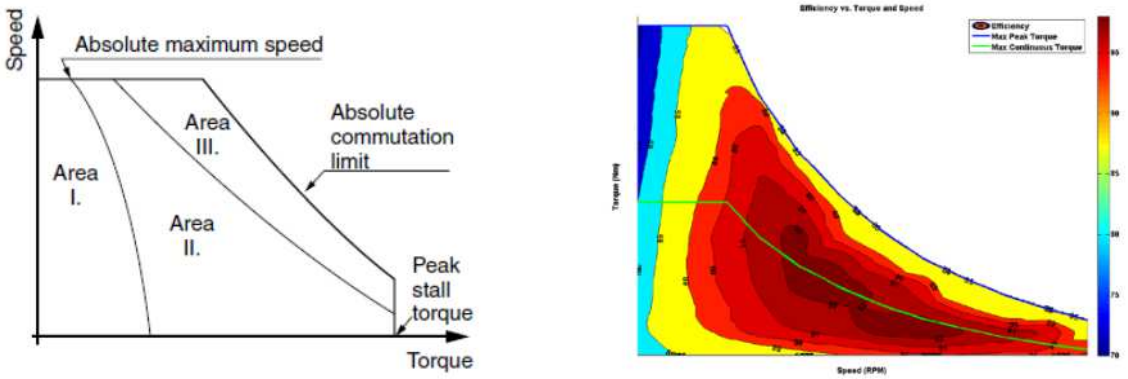


Fig 2.9 Real trend of performance of a BLDC motor

2.4Control

There are different types of control logic that can be selected, for BLDC motors the most widely used is the speed control which is employed in system that receive on/off commands.

A PID controlled gets the calculated speed error from the Halls sensors and checks if it lays on a limited zone, if this error is greater or smaller of respectively the upper a and the lower values of the acceptable values, the PID will generates a step command that will increase variate the tension on the windings according to the needs of the motor. In fact, uch variation will produce an acceleration that is driven by pulse width modulation signal, equivalent to 1 or to 0. This control logic is always active, resulting in a frequency of the signal generation as the graphics shows below:

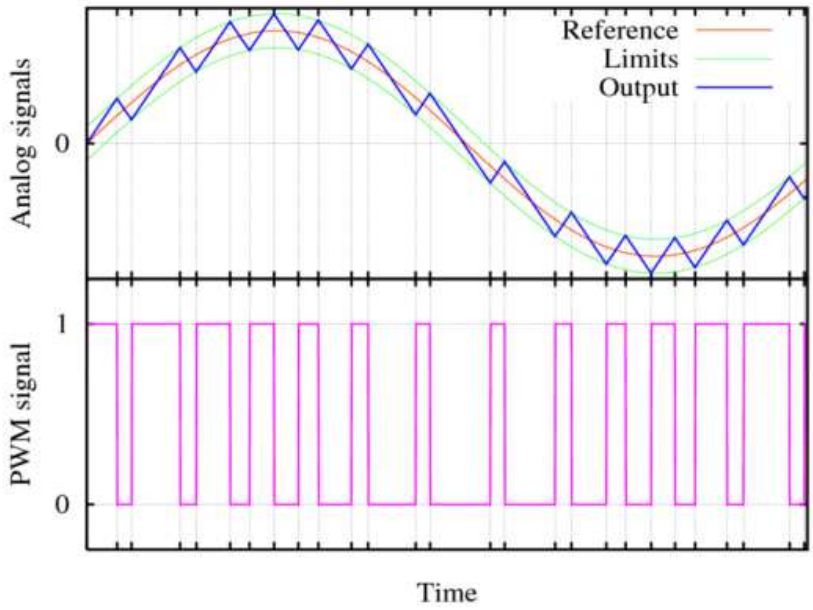


Fig 2.10 PWM control logic

Another control logic that is often applied as an inner loop to the speed control is the torque control, instead of acting on voltage this logic variates the currents on the windings.

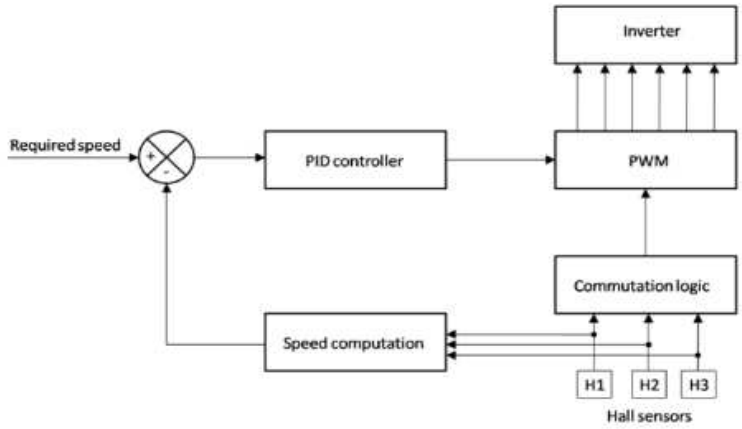


Fig. 2.11 Speed control loop

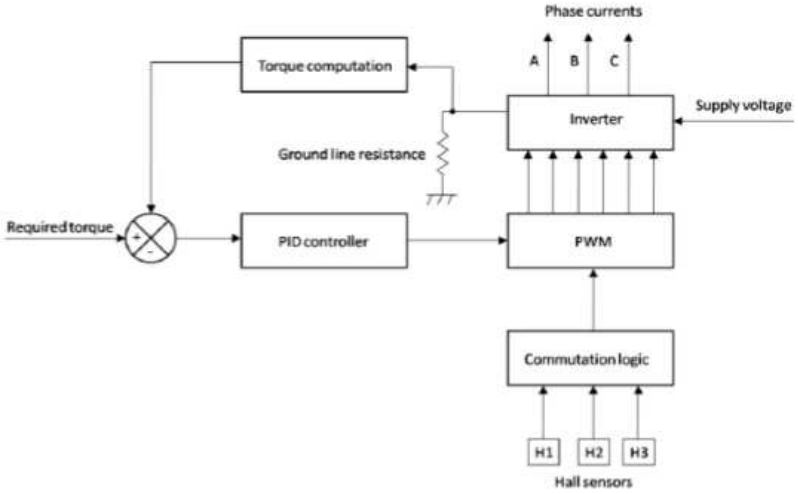


Fig 2.12 torque control loop

3.EMA Models

The reference model and the monitor model are widely described in this chapter, these models were developed in MATLAB simulink and improved during the years.

Parameter	Symbol	Value	Measure Unit
Gain of the Proportional amplifier	G_{prop}	10^5	-
PID controller: proportional gain	GAP	$5 \cdot 10^{-2}$	Nms/rad
PID controller: integrative gain	GAI	0	Nm/rad
PID controller: derivative gain	GAD	0	Nms^2/rad
Maximum power supply voltage	V_{max}	48	V
Maximum current	I_{max}	22.5	A
Maximum motor torque	$T_{m,max}$	1.689	Nm
Torque constant	k_t	0.0752	Nm/A
Back-EMF constant	k_e	0.0752	Vs/rad
Phase resistance	R_s	2.13	Ω
Phase inductance	L_s	$7.2 \cdot 10^{-4}$	H
RL time constant of BLDC motor	τ_{RLs}	L_s/R_s	s
Polar expansions per phase	$2P$	4	-
Number of pole pairs per phase	P	2	-
Current hysteresis band width	hb	0.5	A
Inertial Torque of the motor	J_m	$1.3 \cdot 10^{-5}$	$kg \cdot m^2$
Viscous damping coefficient of the motor	C_m	$\frac{30}{\pi} \cdot 10^{-6}$	Nms/rad
Inertial Torque of the user	J_u	$1.2 \cdot 10^{-5}$	$kg \cdot m^2$
Viscous damping coefficient of the user	C_u	$4.5 \cdot 10^{-7}$	Nms/rad
Static friction torque of the motor	f_{sm}	$0.06 \cdot T_{m,max}$	Nm
Dynamic friction torque of the motor	f_{dm}	$f_{sm}/2$	Nm
Static friction torque of the user	f_{su}	$0.04 \cdot T_{m,max}$	Nm
Dynamic friction torque of the user	f_{du}	$f_{su}/2$	Nm
Nominal backlash	BLK	$5 \cdot 10^{-3}$	Nm

Tab. 3.1 parameters of the models

3.1 Reference model

This model provide a fast and cheap way for the simulation of the EMA, the simulation runs for 0.5 of actual system works by using an integration step equal to 10^{-6} s.

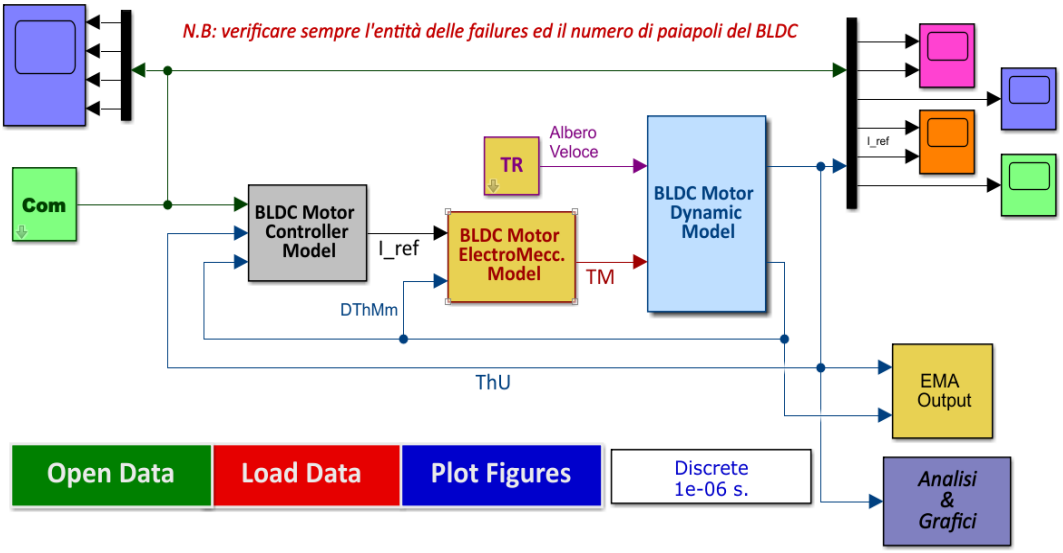


Fig 3.1 Reference model

From the com block comes the input signal,which is an angular position, to the BLDC Motor Controller that closes the loop with by comparing the obtained angular position and speed. The generated error will provided a reference current in input to the BLDC Motor EMA that returns the torque applied by the actuators, than the torque enters the BLDC Motor Dynamic Model where its compared with the resistant torque and the actual speed and position are calculated, these values will be given as the system outputs while also being feedbacked to the controller block.

3.1.1 Com Block

This is the block that generates the command,which can be selected between step,ramp, sine, chirp and a custom one.

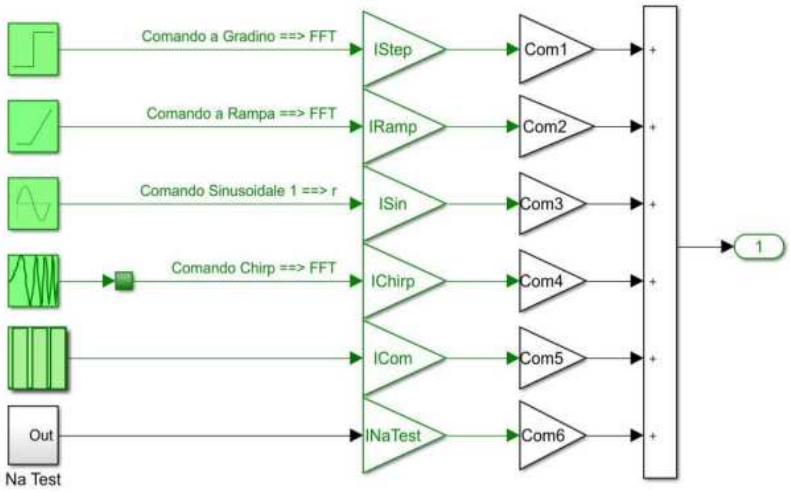


Fig 3.2 Com block

The chirp command has two different modes implemented that can be switched in the small green square, the first uses the parameters coming from the dialogue window of Simulink, the second is hand-made fucntion that decreases from 1 to 0.

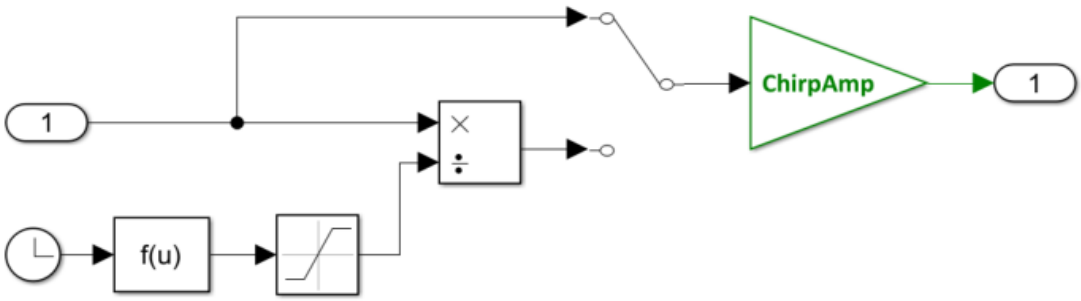


Fig 3.2 Chirp command selection

3.1.2 BLDC Motor Controller Block

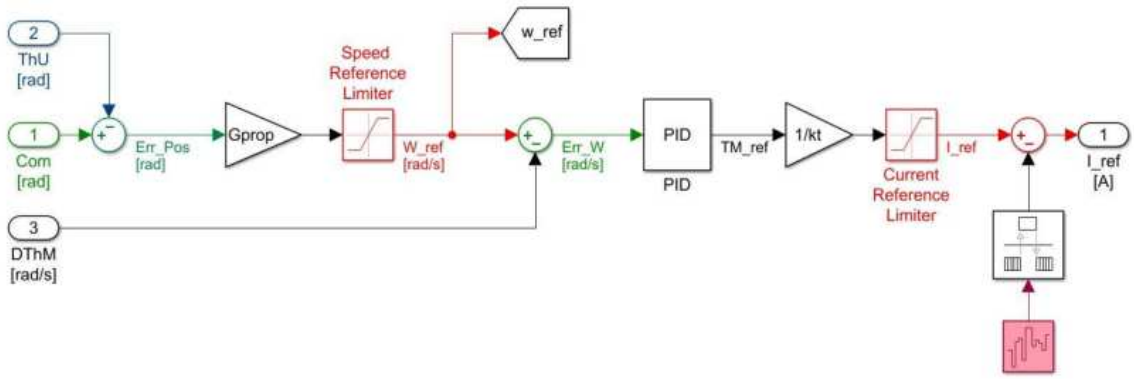


Fig. 3.3 BLDEC Motor Controller Block

This block simulates the controller behavior, an error is generated by the comparison of the command and the feedback. This error is than transformed by the proportional gain in a speed value that is limited by the saturation block(to 8000 rpm) and then compared to the second feedback. The new error enters in the PID block and is transformed into the reference torque, finally the the reference current is obtained by dividing the torque with torque constant. This current, that mustn't be confused with the phases currents, it is just a variable needed to control the motion of the actuation system and is regulated by a saturation block(to 22.5 A). On the last part of the model there is a white noise block block which is not adopted in this work.

3.1.3 BLDC Motor Electromechanical Model Block

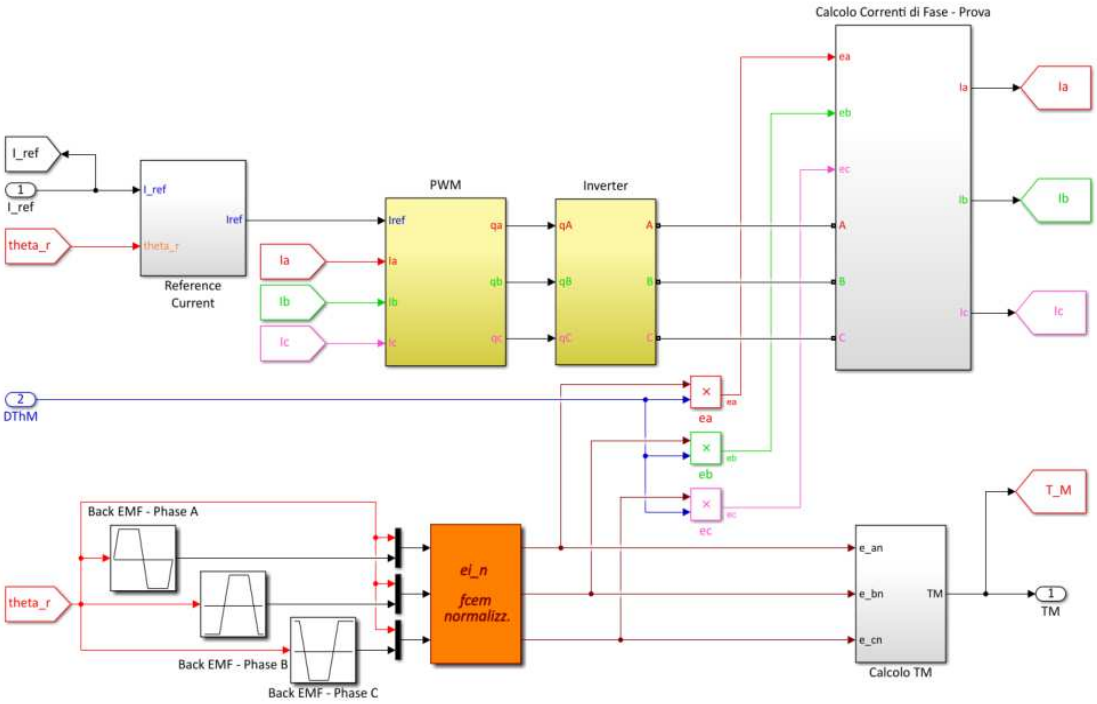


Fig. 3.4 BLDC Motor Electromechanical Model Block

This block describes the effects of the control electronics on the phases of the motors and how the the torque is generated.

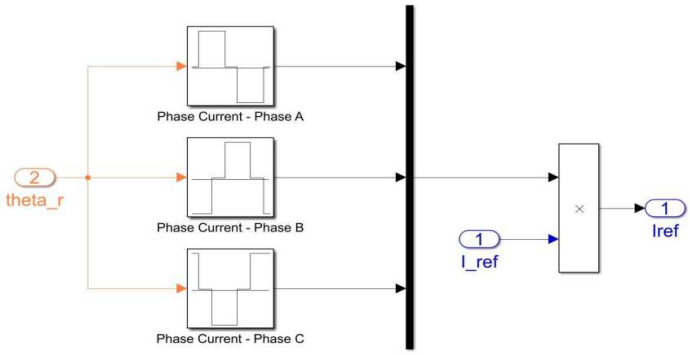


Fig 3.5 Reference current block

First, the reference current is served as an input alongside with the rotor angle to the Reference Current Block.

Inside this subsystem the angle is divided in 3 function that simulates the behavior of the phases of a trapezoidal BLDC motor, those functions gives 1 in case of a positively powered phase (current in), 0 for a turned off phase and -1 for a negatively powered phase (current out). The output is multiplied with the reference current and gives back another reference current that take in account the time trend of the three phases currents.

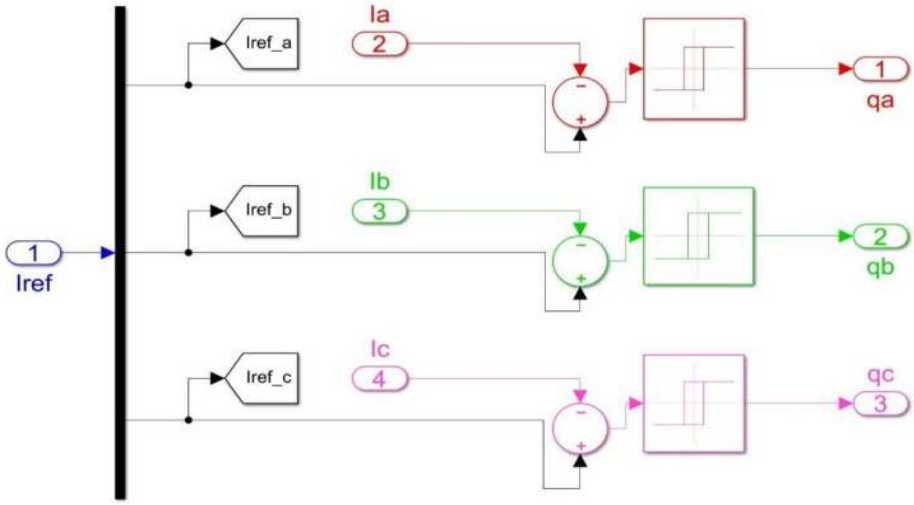


Fig 3.6 PWM block

This value enters in the PWM block where it is compared to the actual currents applied to the windings, the error generated is passed on the hysterics blocks that returns a bolean positive value, 0 if it's lower than -0.5 A or 1 if it's higher than 0.5 A.

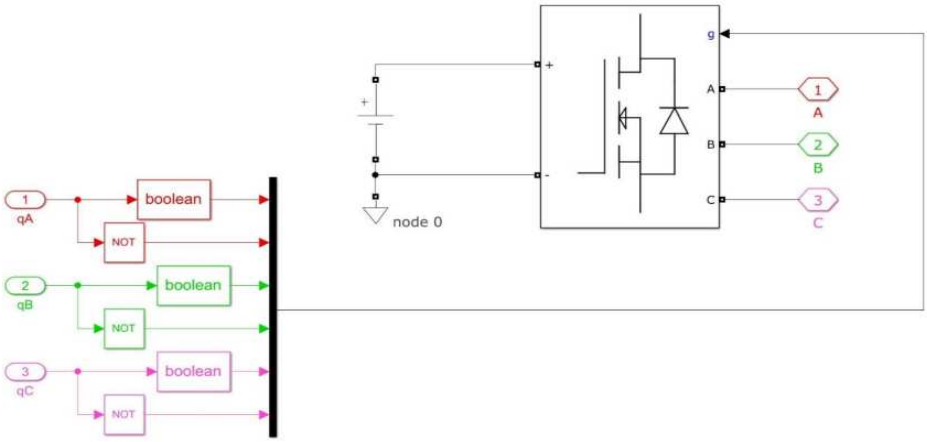


Fig 3.7 Inverter block

The bouleans values enters in the Inverter block and are negated. After that, the original values and the new negated ones reach the H-bridge, this will return the 3 phases voltage as output.

Alongside of the described branch there there is another section of the model where the normalized counter electric force is calculated. The block is called ei_n fcem normalized.

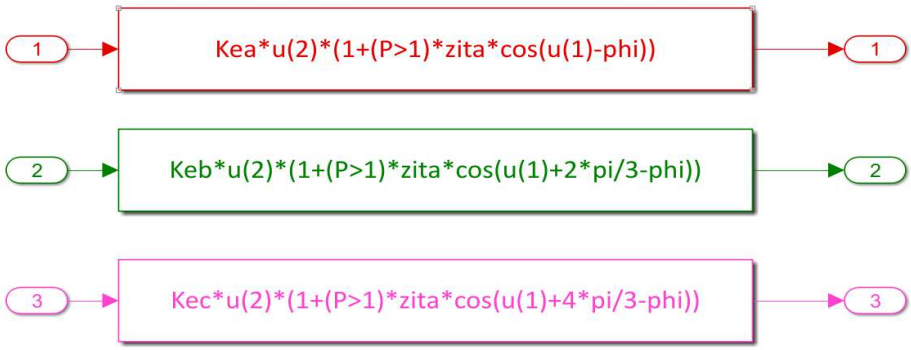


Fig 3.8 normalized CEMF computation block

This part will be later discussed in the implementation of the eccentricity fault.

The outputs of the inverter (tensions of the three phases) and the outputs from the fcem norm. block multiplied with the angular speed of the actuators (gives the true Ke) are served as input to the Phase Current calculation block.

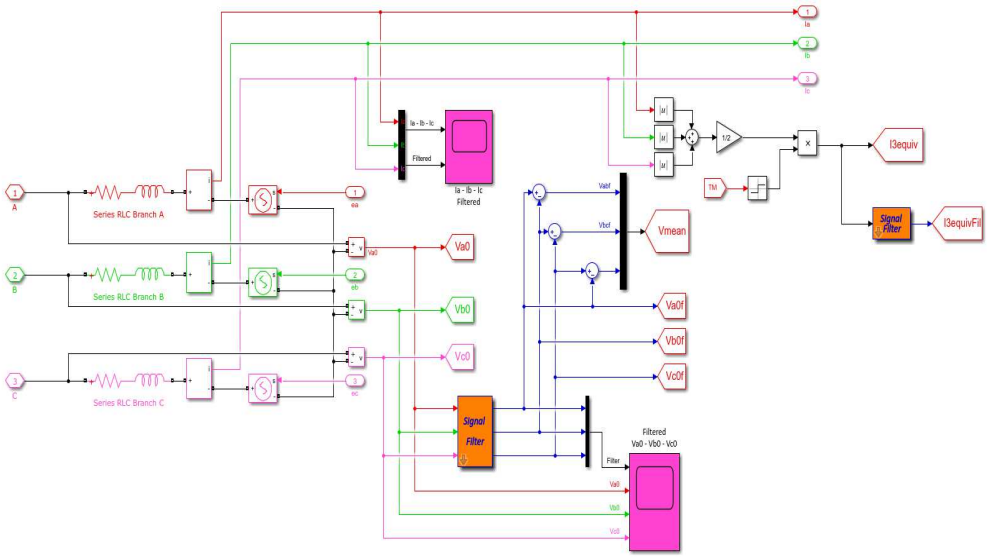
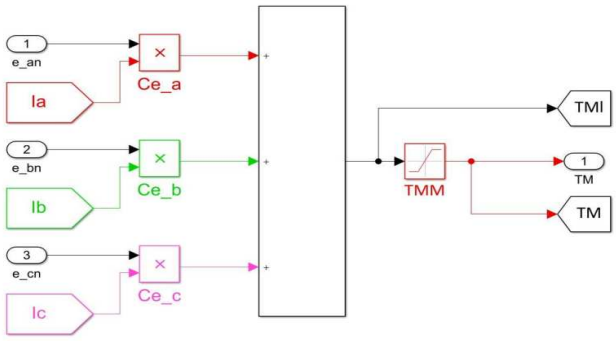


Fig 3.9 current phase calculation subsystem

In order to simulate the star circuit with floating center a SIM power system is added.

This subsystem returns the 3 currents of the phases and a current called I3equiv that is the equivalent single-phase current carrying the same sing of the generated torque and serves as a comparison value for the monitor model output, which is a single current.



3.10 Torque computation block

In the Torque Computation block the torque value is calculated with the normalize kfcem.

3.1.4 BLDC Motor Dynamic Model block

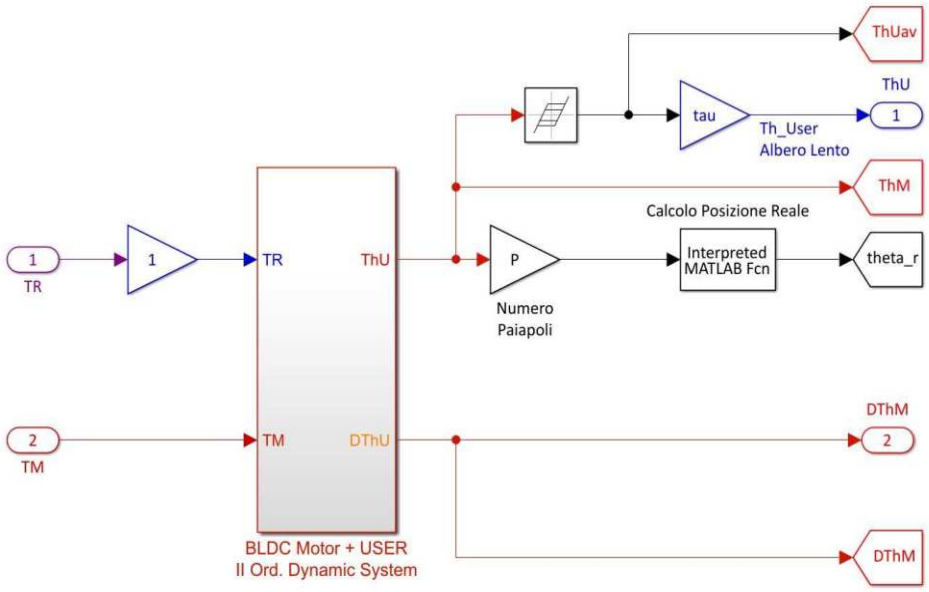


Fig 3.11 BLDC motor dynamic model block

This subsystem take care of the dynamics of the nut-screw actuator linked to the user, a flight command.

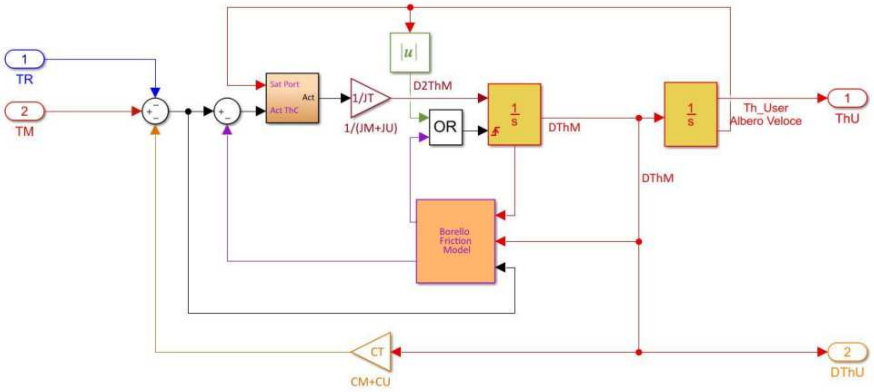


Fig 3.12 Second order dynamic block

First of all, the BLDC Motor + USER block receives in input the calculated torque TM and the resistant torque, the difference of this two parameters is the net value of the torque, this value enters a saturation block that is needed to simulate the mechanical hard-stops of the

servomechanism. This wil give a value that can be 0, if the the end effector is not at one of the limts, +1 or -1 if it is at one limit, depending on the following logic that take in account the direction of the force.

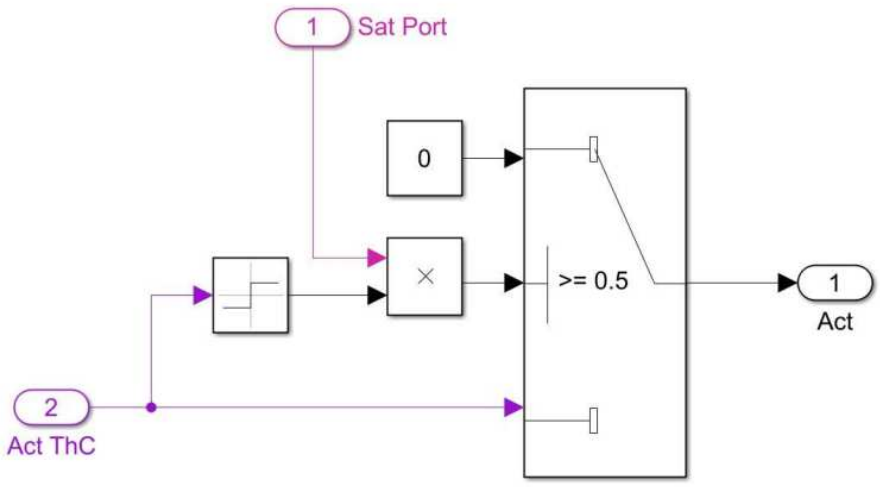


Fig. 3.13 Limit control block

After the saturation, the signal is divided by the the total inertia of the mechanical system and integrated to find speed, then another integration will provide the position. Note that the second integration return a value to the saturation port of the the first integration, this si required to denied further iteration of speed while the system meet a condition of stop and the position is 0. This is regulated by the OR that will gives a +/-1 value if one of the end effectors is reached, switching form 0 to 1 the integration port will cease its action. It takes also in account the friction, which is simulated with the Borello method that is implemented in the block called Borello Friction Model, it is able to simulate both static and dynamic friction by regulating the transition between this two phase of friction, which is a non linear phenomena.

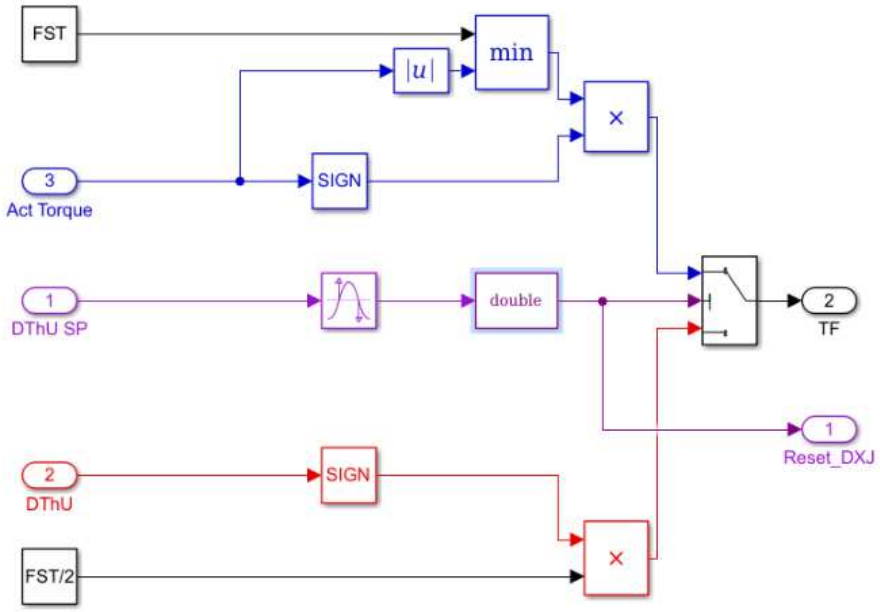


Fig 3.13 Borello friction

After the USER block, the signals are processed by tacking account of the backlash. The outputs are the real position and speed of user and motor.

3.2 Monitor model

The simplest model, it simulates a single-phase actuator and is suited for application that requires faster computational time. This model, paired with an optimization algorithm, should be able to help in the fault detection of the real actuator.

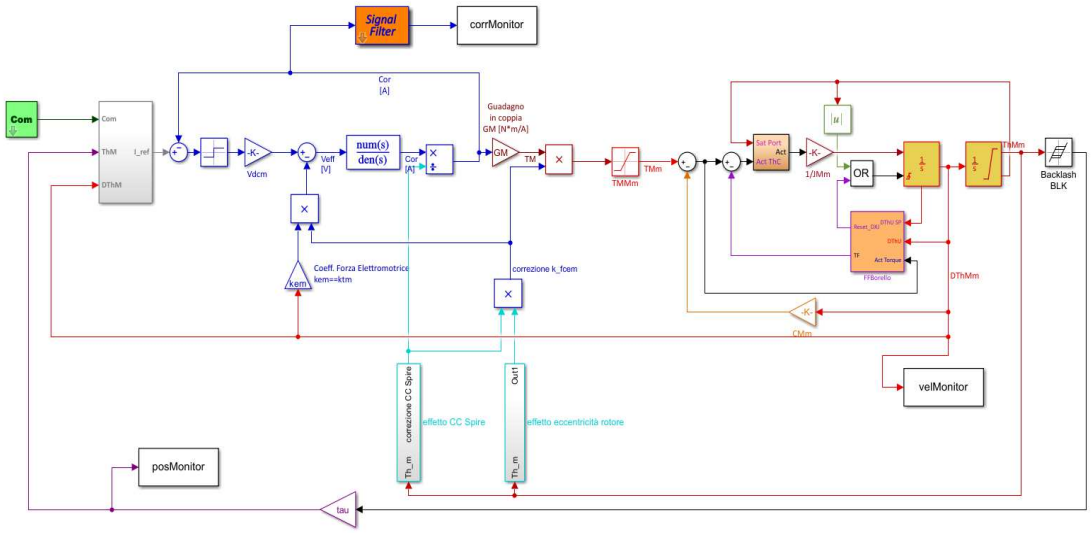


Fig 3.14 Monitor model

The command block is the same as the RF.

3.2.1 Control block

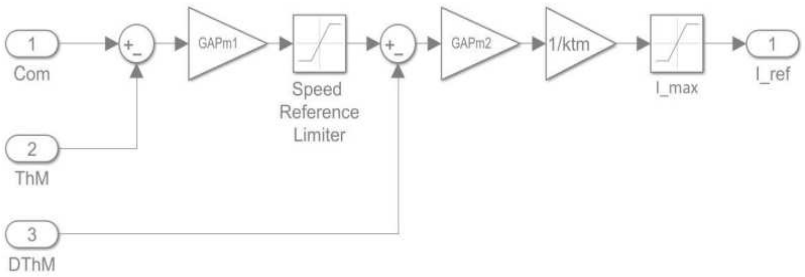
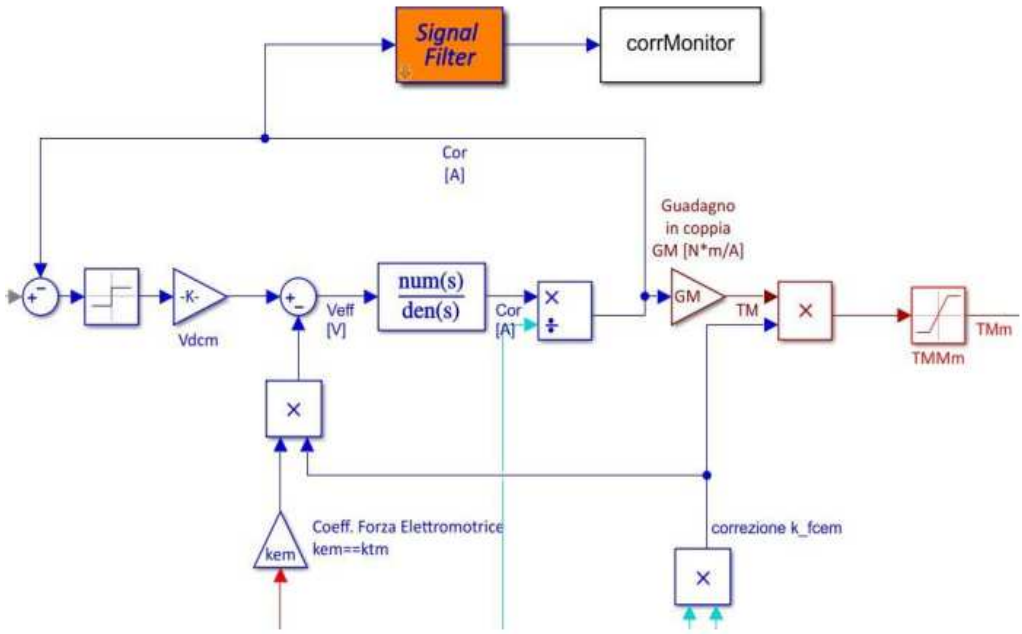


Fig. 3.15 control block for MM

The control block has a similar structure to the RF counterpart, it is simplified by the substitution of the PID with another gain. Also, the white noise block is absent.

3.2.2 Electromechanical model



3.16 single-phase BLDC subsystem

The error between currents enters the sign block; being a single-phase actuator the tension can be only positive or negative, the sign block will take care of this behavior by returning +1,0 or -1 for output that is multiplied with the with the nominal inverter tension (48 V) to calculated the true value of voltage applied to the rotor. This voltage is subjected to loses that will be illustrated in the next chapter, than it is commuted with the first order transfer function:

$$T_f = \frac{\left(\frac{1}{R_m}\right)}{\left(\tau_{RLm}+1\right)}$$

where R_m is the resistance of the winding and τ_{RLm} is the ratio between resistance and inductance

After, the current is multiplied with the torque gain to calculates the actuators torque.

3.2.3 Mechanical part

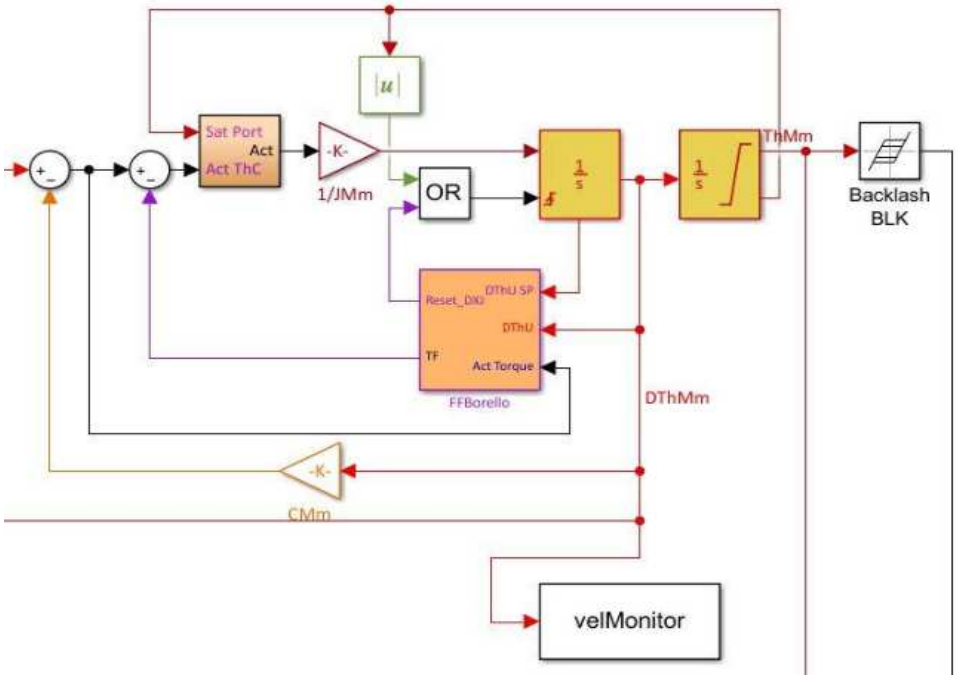


Fig. 3.17 Mechanical part of the MM

The mechanical parts is equivalent to the RF model one.

4.Faults

As mentioned before, the EMA actuator has been introduced in the aeronautics field recently and it is applied to secondary flight controls for reliability issues, in particular lack of knowledge about it's faulty modes that can be quite unpredictable if combined in a multi-modal fault situation. Still, each fault is well know and the 4 main categories are listed here:

- Sensor faults: an error in the process of feedback signal acquisition from sensors, this can modify the dynamic response of the systems driving by a modification of the control laws that are governing the actuator. They are divide in bias,scaling and drift faults.
- Electrical or electronics faults: these faults are similar for all the electrical systems onboard, the are different type of factors that can produce this kind of faults, for example: problems of electromagnetic compatibility, particles that can produce electrical arcs, overheating of the devices.
- Mechanical and structural faults: the most important category, these faults interests the reducer and the ball screw actuator, they can be caused by excessive load, corrosion, manufactory defects or lack of lubrification.
- Motor faults: the BLDC motor works at high angular speed, this can stress the entire motor structure and the friction causes frequent overheating.

The following tables shows the various failures for each component:

Component	Fault	Failure	Relative probability	Relative criticality
Connectors	Degraded operation (increase of resistance)	Disconnect	5	6
	Intermittent contact	Disconnect	3	7
Stator	Stator coil fails open (results in degraded EMA performance)	Opening failure	4	4
	Insulation deterioration/wire chafing (reduced or intermittent current through stator coil or intermittent short	Short-circuit	5	5
Resolver	Coil fails open (can result in inaccurate position reports)	Opening failure	4	10
	Intermittent coil failures	Permanent coil failure	5	7
	Insulation deterioration/wire chafing	Short-circuit	5	7
Rotor and magnets	Rotor-magnets chemical bond deterioration	Complete magnet separation, likely leading to motor failure	2	10
	Rotor eccentricity	Bearing support failure	3	6

Tab. 4.1 Mechanical and structural fault modes

Component	Fault	Failure	Relative probability	Relative criticality
Screw	Spalling	Severe vibrations, metal flakes separating	5	3
	Wear/backlash	Severe backlash	7	3
Nut	Spalling (mild)	Severe vibrations, metal flakes separating	5	3
	Backlash	Severe backlash	7	3
	Degraded operation	Seizure/disintegration	3	5
	Binding/sticking	Seizure/disintegration	3	3
	Bent/dented/warped	Seizure/disintegration	1	5
Ball returns	Jam	Seizure/disintegration	5	8
Bearings	Spalling	Severe vibrations, metal flakes separating	5	3
	Binding/sticking	Seizure/disintegration	2	4
	Corroded	Severe vibrations, metal flakes separating, seizure/disintegration	2	5
	Backlash	Severe backlash, vibrations, disintegration	7	3
Piston	Crak(s), slop/play	Structural failure	1	10
Dynamic seals	Wear	Structural failure	4	6
	Structural failure	Structural failure	3	8
Static seals	Structural failure	Structural failure	2	8
Balls	Spalling/deformation	Severe vibrations, metal flakes separating	5	3
	Excessive wear	Backlash	7	5
Mountings	Crack(s), slop/play	Complete failure	1	7
Lubricant	Contamination	Seizure/disintegration	8	5
	Chemical breakdown	Seizure/disintegration	4	5
	Run-dry	Seizure/disintegration	3	10

Tab. 4.2 motor fault modes

Component	Fault	Failure	Relative probability	Relative criticality
Power supply	Short-circuit	Short-circuit	5	10
	Open circuit	Open circuit	5	10
	Intermittent performance	Short-circuit or open circuit	5	8
	Open circuit	Open circuit	5	10
	Thermal runaway	Dielectric breakdown of components, leading to open or short-circuit	6	10
Controller capacitors	Dielectric breakdown	Short-circuit or open circuit	4	8
Controller transistors	Dielectric breakdown	Short-circuit or open circuit	4	8
Wiring	Insulation deterioration/wire chafing	Short-circuit or open circuit	5	8
	Short-circuit	Short-circuit	5	10
	Open circuit	Open circuit	5	10
Solder joints	Intermittent contact	Disconnect	5	8

Tab 4.3 Electrical fault modes

4.1 Noise

As seen before, this fault is implemented in form of white noise in the BLDC motor controller block on the reference model while it is absent on the monitor model. This choice was made by considering the fact that a monitor model should provides information on the faulty system even if noise occur.

Noise is a disturbance that affects signals acquisition, it can be acoustic, electromagnetic,electrostatic, channel or processing noise. It can be also classified by frequency, there is white noise, limited-band white noise,color noise, narrow band noise,impulsive noise and transient noise.

The white noise blocks generates random normal distributed numbers that are multiples of the integration steps and thus can affect the signals.

For this work it is not considered in the simulation of reference model.

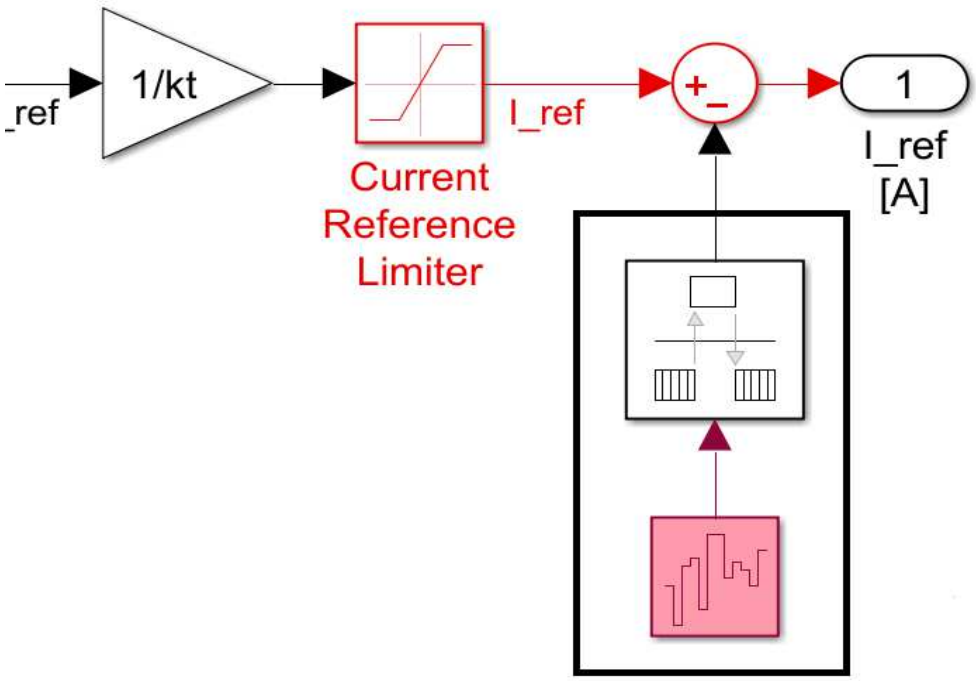


Fig 4.1 Noise implementation in RM

4.2 Friction

The friction is always present and can increase for various factors. This increment will cause a loss in efficiency caused by the added power needed to move the components, it will also create an excessive structural and thermal stress to the mechanical parts, therefore is very important to detect and correct in time.

The Coulomb model, already showed in the Borello block, approximates both static and dynamic friction, it also regulates the transition from the first to the second that follows a non linear behaviour by adopting the following process:

- Identification of the rotation with the selection of the right sign of the friction torque.
- Computing the torque, taking care of the load applied .
- Selection of the condition,static or dynamic.
- Verified of eventual mechanical stops.
- Computing the break away from a former standstill mechanical part.

$$F_f = \begin{cases} F_{act}, & \dot{x} = 0 \cap |F_{act}| \leq F_{sj} \\ F_{dj} \cdot sign(F_{act}), & \dot{x} = 0 \cap |F_{act}| > F_{sj} \\ F_{dj} \cdot sign(\dot{x}), & \dot{x} \neq 0 \end{cases}$$

F_{act} is the load on the system, F_{sj} is the static friction , F_{dj} is the dynamic friction,F_f is the computed friction force.

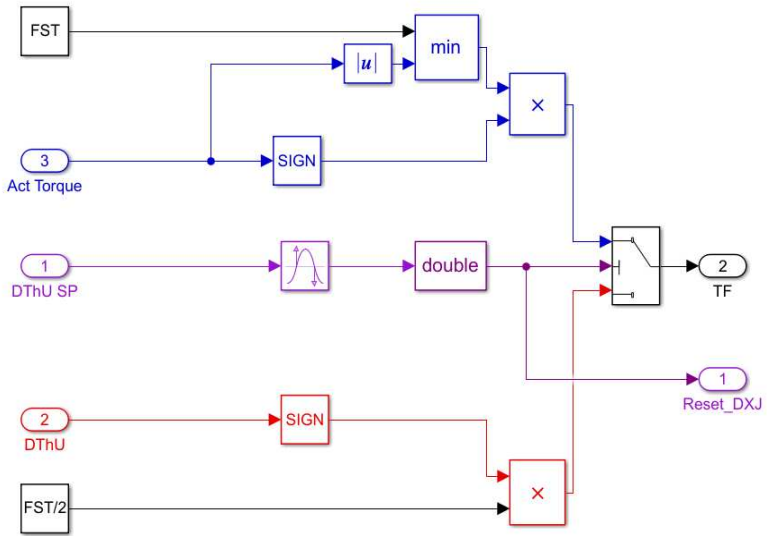


Fig 4.2 Borello friction

4.3 Backlash

The motion transmission is achieved with contact between mechanical parts and the system is subjected to degradation caused by friction, stress and other issues. To prevent the insurgency of these problem, a tiny gap between transmission components for the lubricant oil is always present.

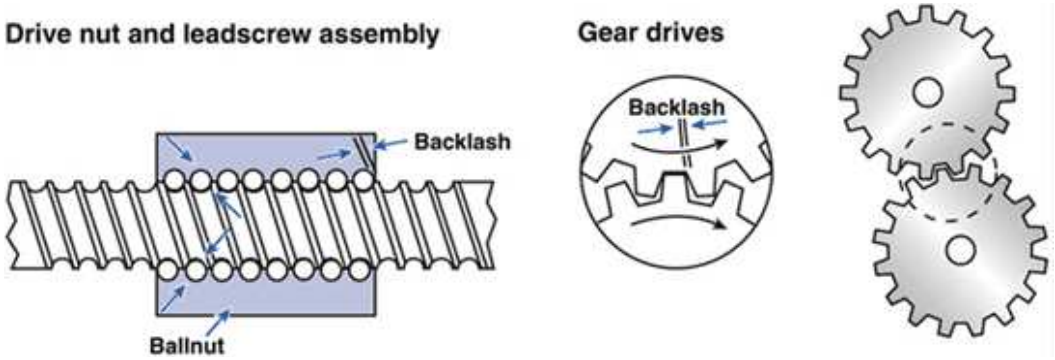


Fig 4.3 Backlash

This gap will eventually grow during the operating life of the actuator, causing power loss to the the transmission and reactivity problems to the user rotation. In the EMA models, the backlash is modelled for the ball screw tacking in account the relative motion between the ball-nut and the screw.

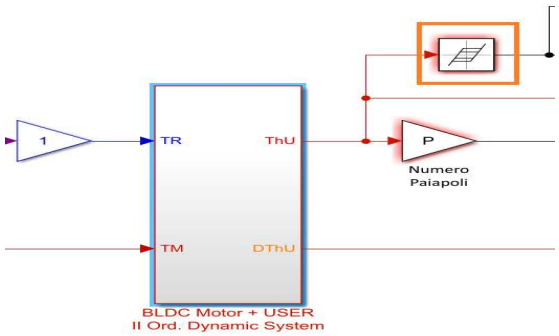


Fig 4.4 backlash implementation in RM

The implementation of the backlash is simply made with the Backlash block from the Simulink library, it is positioned after the modeled mechanical part in each model.

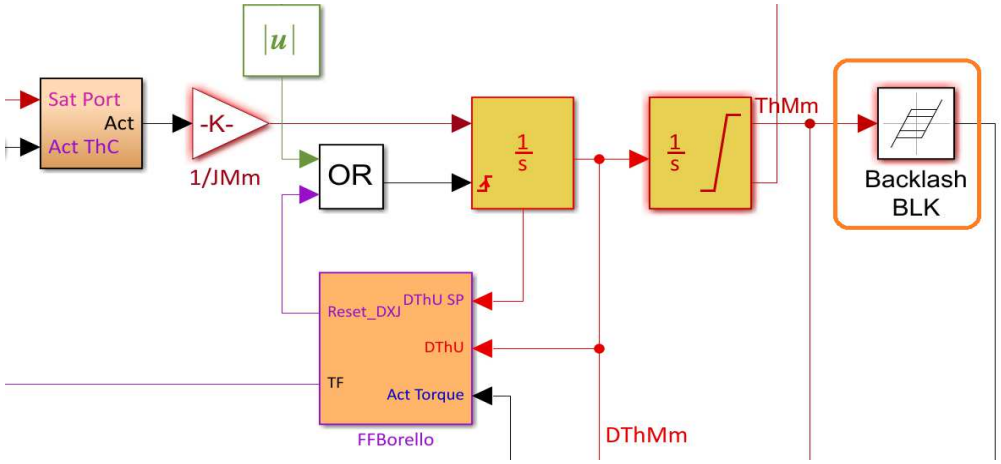


Fig 4.5 Backlash implementation in MM

4.4 Short circuit

Usually this fault starts with a coil to coil short circuit, generated between the winding of the same phase that came in contacts due to the degradations of the insulating material between them caused by overheating during the operative life of the motor. This phenomena brings to a reduction of both resistance and inductance that will increase the current with the same tension, creating more stress and overheat.

Eventually, the short circuit will propagates between windings of different phases or between winding and the stator steel, causing the failure of the entire actuation system.

The coil to coil mode is the most important, hence it is the only implemented in the model.

On the reference model it is possible to change the percentage of short circuit DIRECTLY on each one of the phases, as the inductance decreases the force will decrease too:

$$K_{ei} = K_e \cdot N_i$$

where K_{ei} is the counter electromotive coefficient.

$$R_{ij} = \frac{R_s}{(2 \cdot (N_i + N_j))}$$

$$L_{ij} = \frac{L_s}{(2 \cdot (L_i^2 + L_j^2))}$$

where R_s and L_s are the phase to phase resistance and inductance while R_{ij} and L_{ij} are the faulty ones.

$$R_i = \frac{R_s}{(2 \cdot N_i)}$$

$$L_i = \frac{L_s}{(2 \cdot L_i^2)}$$

where R_i and L_i are the coil to coil resistance and inductance on faulty conditions.

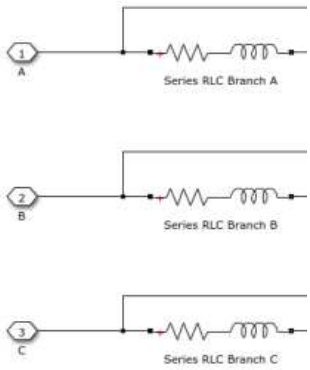


Fig 4.5 three-phase current implemented in the RM

The implementation on the reference model follows the same principle of the calculated current I_{3equiv}, hence the short circuit is calculated as an avergae of the three coefficients:

$$N_{equi} = \frac{(N_a + N_b + N_c)}{3}$$

This is mandatory since it is impossible to locate the faulty phase with the monitor model.

The other parameters are calculated as:

$$R_{equi} = R_{equi1} \cdot N_{equi}$$

$$L_{equi} = L_{equi1} \cdot N_{equi}^2$$

$$k_{fecem} = k_{fecem1} \cdot N_{equi}$$

$$G_{Mequi1} = G_{Mequi1} \cdot N_{equi}$$

where the $_{equiv1}$ stands for the nominal condition.

The rotor angular position is used to modulate the characteristic of the motor, to avoid issues related to the carrier frequency of short circuit and eccentricity.

$$f(\theta_m) = \begin{cases} \dfrac{N_b + N_c}{2}, & \text{if } -\dfrac{\pi}{6} < \theta_e < \dfrac{\pi}{6} \\ \dfrac{N_a + N_b}{2}, & \text{if } \dfrac{\pi}{6} < \theta_e < \dfrac{\pi}{2} \\ \dfrac{N_a + N_c}{2}, & \text{if } \dfrac{\pi}{2} < \theta_e < \dfrac{5\pi}{6} \end{cases}$$

Since this modulation takes care of only two phases, a correction is implemented as followed:

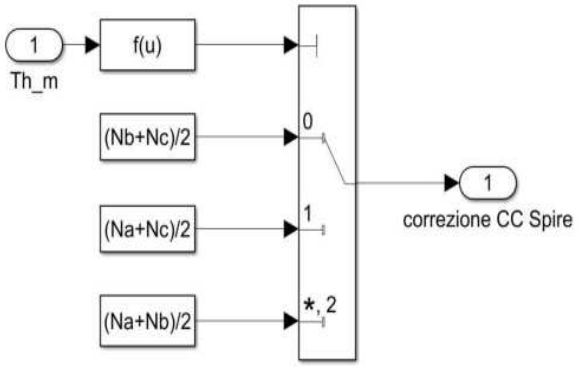


Fig 4.6 modulating function

4.5 Eccentricity

An eccentricity is a fault that happens when the air gap between the rotor and the stator changes due to a difference in the center of symmetry of one, or both, of those components. The air gap is then dissimilar for each angular position, with a different distance from each point of the stator and rotor that will gradually damage the motor.

This fault is always present, since it is not possible to create a perfect air gap, and increases during the working life of the motor thanks to vibration and usury of the bearings.

There are illustrated three cases of eccentricity:

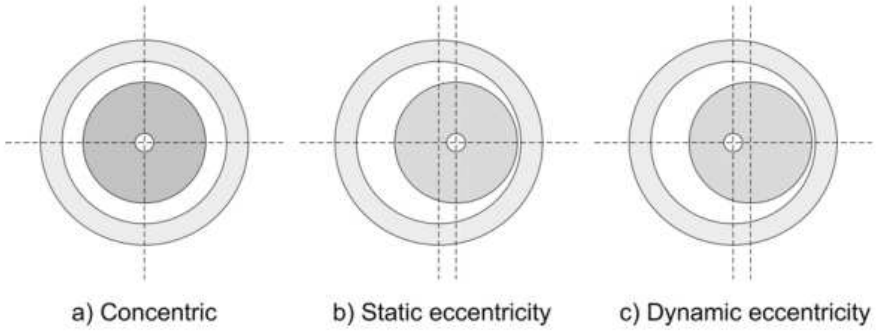


Fig. 4.7 Types of eccentricities

Only the static eccentricity is studied in this work, with reference to the figure below :

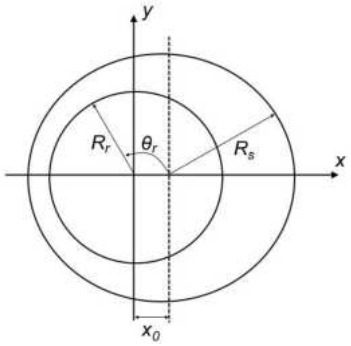


Fig 4.8 Reference system for air gap definition

Assuming both rotor and stator as rigid bodies it is possible to obtain the air gap with the following passages:

$$x^2 + y^2 = R_r^2$$

is the equation of the rotor circumference.

$$(x - x_0)^2 + y^2 = R_s^2$$

is the equation of the stator circumference.

With the polar coordinates:

$$\rho = R_r$$

$$\rho^2 - 2\rho\, x_0 \cos(\theta_r) - R_s^2 + x_0^2 = 0$$

by approximating it with a Taylor series of a second order, the air gap is:

$$g = g_0(1 + \zeta \cdot \cos(\theta_r))$$

with $g_0 = R_s - R_r$ as the air gap in non faulty condition and $\zeta = \frac{x_0}{g_0}$ as the ratio between the misalignment and the nominal air gap.

The magnetic flux is affected and the difference between forces modules creates an imbalance in the system:

$F_{mm} = \Phi \cdot R$ is the magnetomotive force,where $R = \frac{l}{(\mu_r \mu_s S)}$ is the reluctance, S is the surface of the rotor permeated by the magnetic flux.

The magnetic flux can be rewritten as a function of the air gap:

$$\Phi = \frac{(F_{mm} \mu_0 S)}{(g \cdot (\theta_1 + g(\theta_1 + \frac{\pi}{P})))}$$

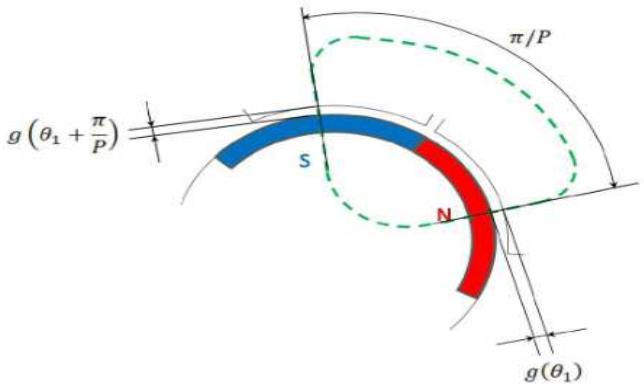


Fig. 4.9 Magnetic circuit through the air gap

For the reference model it is implemented in the computation of the Kfcem:

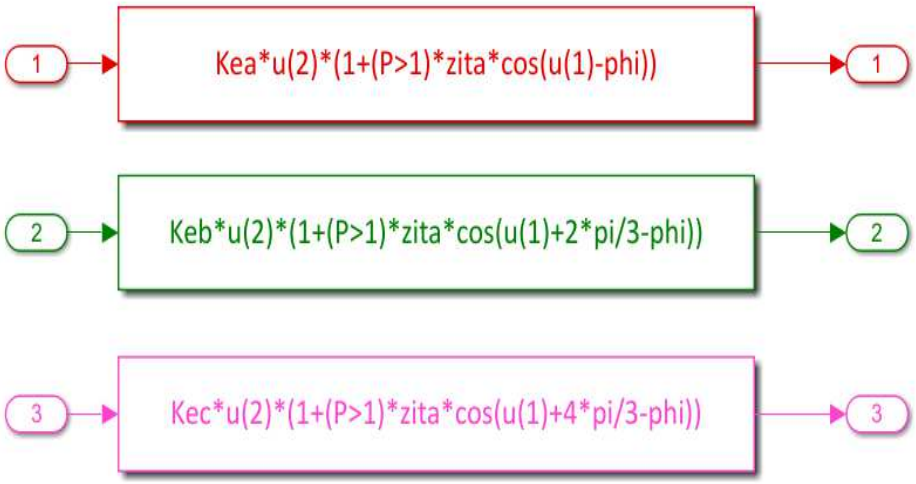


Fig 4.10 CEM F computation on Simulink

in this way, both torque gain and counter electromotive force gain depends on the angular position of the rotor.

In the monitor model ζ is normalized in a interval from 0 to 0.42, the new variable is called Z.

The calculation of the effect on the K_{fecm} is effectuated by the dedicated block.

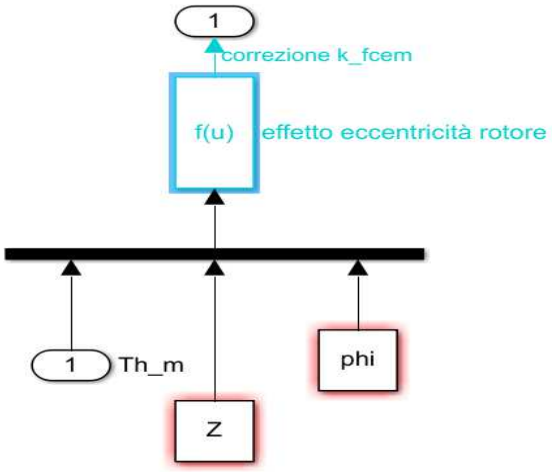


Fig 4.11 Eccentricity modification block

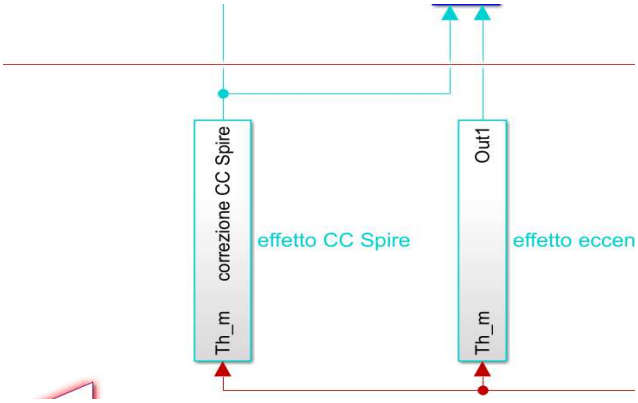


Fig. 4.12 Effect of eccentricity implement on MM

The computed equation is:

$$K'_{fcem} = K_{fcem}\left(1 - Z\cdot\left(\cos\left(P\,\theta_m - \phi\right) + sawthoot\cos\left(6P\,\theta_m - \pi\right)\cdot\sin\left(P\,\theta_m - \pi\right)\right)\right)$$

4.6 Proportional gain

There are many electronic faults that can be occurs during the working life of the actuators, caused by different issues such as overheating, loss of power in the line and possibles short circuit on critical elements. Most of these failures follows a "cascade" behaviour , starting with a failures that eventually propagates and generates other failures in the system until it completely breaks.

For these reason, a generic and progressive electronic fault is studied in this work and its faulty behaviour is simulated in the controller block.

This fault is implemented in both models with a simple multiplication of the proportional gain with a parameter G that is limited between 0.5 and 1.5, to simulates both loss and increases of gain from nominal conditions.

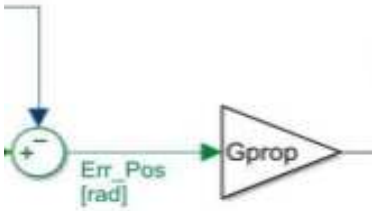


Fig. 4.13 Gain implemented on RM

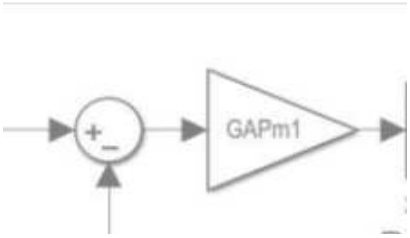


Fig. 4.14 gain implemented on MM

5.Dragonfly algorithm

5.1 Introduction to optimization algorithms

The optimization algorithm in literature are divided in two main classes:

- Deterministic: this kind of algorithm are also defined exact methods that, by using equations searches the space of solutions and reach the global minimum by following logical and well defined steps. These methods requires a huge computational time, paired with growing complexity of the problems on which they are applied, are no longer suitable for all engineer applications.
- Stochastic and heuristics: these methods are based on a random search lead by a group of agents or a single solution, in every iterations the algorithm tries to improve the state of the solutions and bring them to convergence. Running multiple optimization will lead to different results, the better methods should be able to run multiple times and achieve similar good results.

Stochastic and meta-heuristic methods can be also divided in:

- Evolutionary: the most developed algorithm so far, these are nature based methods that reproduces the life cycle of a population starting from birth, development, grown, reproduction and selection, this last one is often applied to the fittest members of the population and a limited random part. The initialization starts with a few agents that in the course of the optimization evolves in order to find the better solution. In the previous works, which this one is the continuation, the Genetic Algorithm and the Differential Evolution belongs to this particular type.
- Swarm intelligence: a more recent class f optimization methods that mimic the behaviour of a certain class of organisms performing a particular task. The search agents moves in the domain of possible solutions and compute the fitness function multiple times, the result of the function is then confronted and its value triggers the equations that regulates

the motion of the swarm,the whole processes is than repeat form iteration to iteration until convergence or other requirements are meets. These methods are often utilized with parallelization so the computational time is better than the evolutionary class,also , the reliability is higher because they depends on less parameters. In other works, Particles Swarm optimization and Grey Wolves optimization are an example of such methods, and the DA algorithm belong to this class too.

The No Free Lunch theorem states that if an algorithm works well with a particular class of problems, it will perform poorly on another kind of class. There aren't perfect methods, each one should be applied in order to find the reliable.

5.2 Procedure for the application of a general optimization algorithm

As mentioned before, the aim of this work is to test a particular optimization algorithm applied to the linear and simplest model of the EMA.

This application is divided in multiplied steps:

- The reference models parameters are initialized.
- The faulty behavior is chosen by the user, it can be a single-fault or multi-faults case.
- The simulation of the EMA reference starts performing the behavior of the system for 0.5 second.
- An input from the user will select the algorithm and its main variables.
- The monitor model starts and run for multiple times, from each iterations the algorithm will run.

5.3 Introduction to the dragonflies algorithm

The inspiration for DA comes from the hunting behavior of a dragonflies swarms in nature, which is divided in a dynamic and a static phase. This is modeled and recreated by the exploration and exploitation phases of the optimizations, in this two blocks the minimal value of the fitness function is searched with equations that mimicking different pattern of the hunts, such as food attraction, cohesion in navigation and others. In particular, this kind of operations are regulated to transition from the exploration phase to the exploitation phases. This transition is crucial , in the dynamic swarming multiple groups of solutions are subdued to these corrective patterns in order to find a local minimum, than the static swarming has to reach the global minimum with precision.

5.3.1 Primitive corrective patterns between individuals of a swarm

The algorithm has five primitive corrective, each one of these is applied to a limited sets of solutions circumscribed in a radius, that changes with the number of iterations by following the equation:

$$r = \left(\frac{ub - lb}{4}\right) + \left((ub - lb)\left(\frac{iteration}{MaxNumber}\right)2\right)$$

where ub and lb are respectively the upper and lower boundary, in this work they are set to 1 and 0 and so the equation becomes:

$$r = 0.25 + \left(\frac{iteration}{MaxNumber}\right)2$$

The radius is a property of each single dragonfly that is increased during the optimization to adjust the transition between dynamic and static swarming.

The corrective patterns are:

- Separation, which defines the static collision avoidance behavior of an individual from each other individuals in the selected neighborhood.

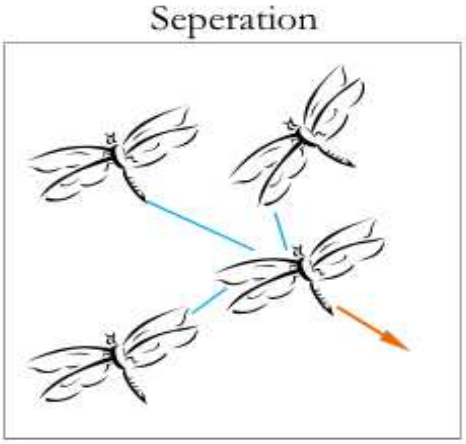


Fig 5.1 Separation pattern

The separation is calculated as:

$$S_i = -\sum_{j=1}^{N} X - X_j$$

Where X is the position of current individual, Xj shows the positions of neighboring individuals and N is the number of neighboring individuals.

- Alignment, it represents the individual tendency of matching the velocity with the neighboring individuals.

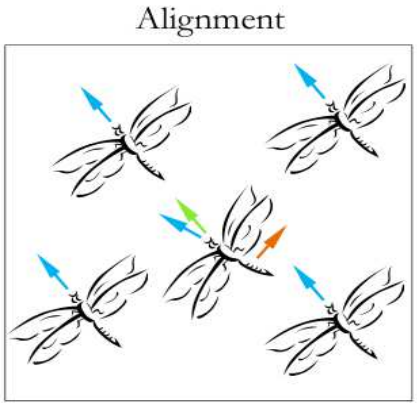


Fig 5.1 Alignment pattern

This is calculated as:

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N}$$

Where Vj is the velocity of the j-th neighboring individual.

- Cohesion, which refers to the trend of individuals in moving themselves towards the center of mass of the whole neighborhood.

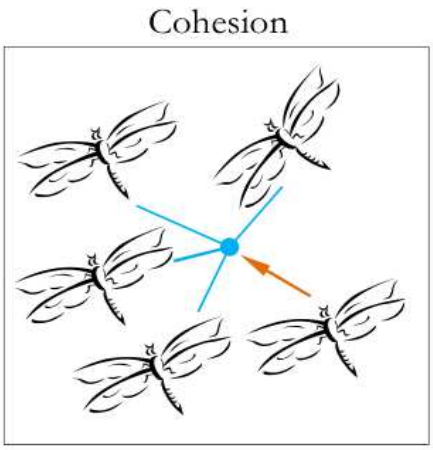


Fig. 5.3 Cohesion pattern

The coheiosion is calulated with:

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X$$

Where X is the position of current individual, Xj shows the positions of neighboring individuals and N is the number of neighboring individuals.

- Attraction to food, the fist operator that mimic survival the survival behaviour that lead the swarm towards the aviable food source

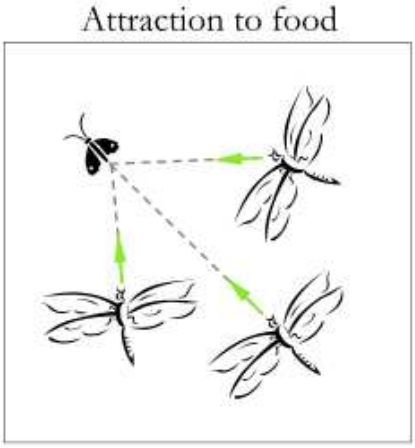


Fig 5.4 Attraction to food

Is calculated as:

$$F_i = X^f - X$$

Where X is the position of the current individual and X^f is the position of the food.

- Distraction form enemy, which lead the swarm away from dangers in order to survives.

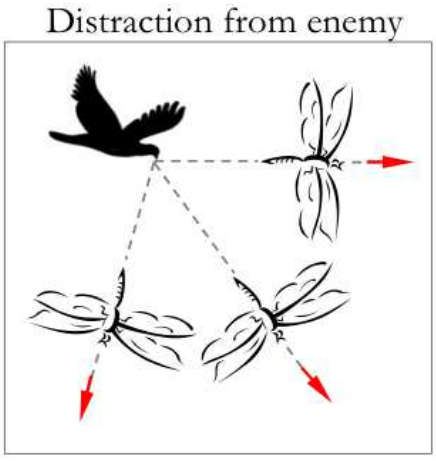


Fig 5.5 Distraction from enemy

The distraction is calculated as:

$$E_i = X^e - X$$

Where X is the position of the current individual and X^e is the position of the enemy.

The first three operators are weighed in a proper way that will help the transition between exploration to exploitation, the dragonflies tend to move with high alignment while maintaining proper separation and a low level of cohesion during the static phase, instead, by moving in dynamic hunt phase the swarm will gain high level of cohesion and low levels of separation and alignment.

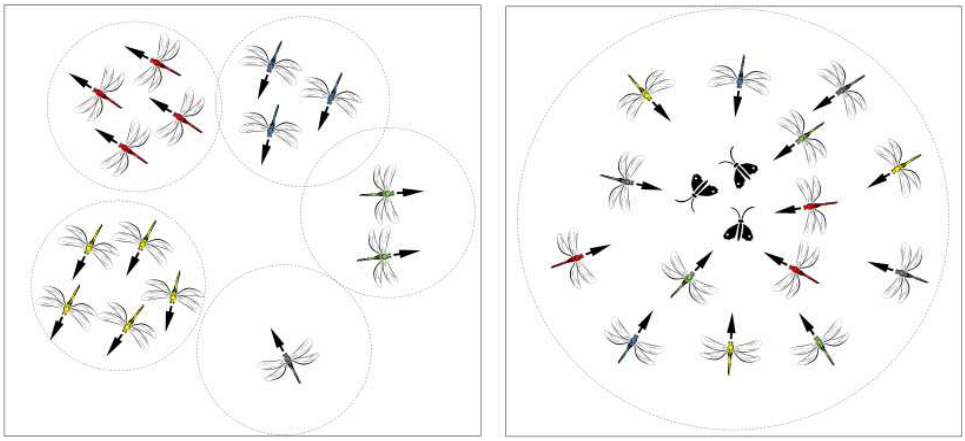


Fig 5.6 exploration and exploitation

The food attraction increases during the optimization to mimic the final stage of the hunt and lead the solutions to the global minimum, while the enemy distraction operator is able to avoid the worst solutions that are increased multiple times to get them out of the radius range of the rest of the swarm.

All this parameter works together to hunt the best solution and are exploited in these equations:

$$\Delta X_{t+1} = sS_i + aA_i + cC_i + fF_i + eE_i + w\Delta X_t$$

hence, $$X_{t+1} = X_t + \Delta X_{t+1}$$

If the choosen dragonfly doesn't have any neighbouring individuals it is required to randmly fly in the search space by following the equation of the Lévy flight:

$$Lévy(x) = \frac{0.01 \cdot r_1 \cdot \sigma}{|r_2^{\frac{1}{\beta}}|}$$

where r_1 and r_2 are random numbers from [0,1] , β is a constant (for this work is set to 1.5) and σ is calculated with:

$$\sigma = \frac{\Gamma(1+\beta) \cdot \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \cdot \beta \cdot (2^{\frac{\beta-1}{2}})}$$

where $\Gamma(x) = (x-1)!$

this possibility increases the stochastic behavior of the algorithm.

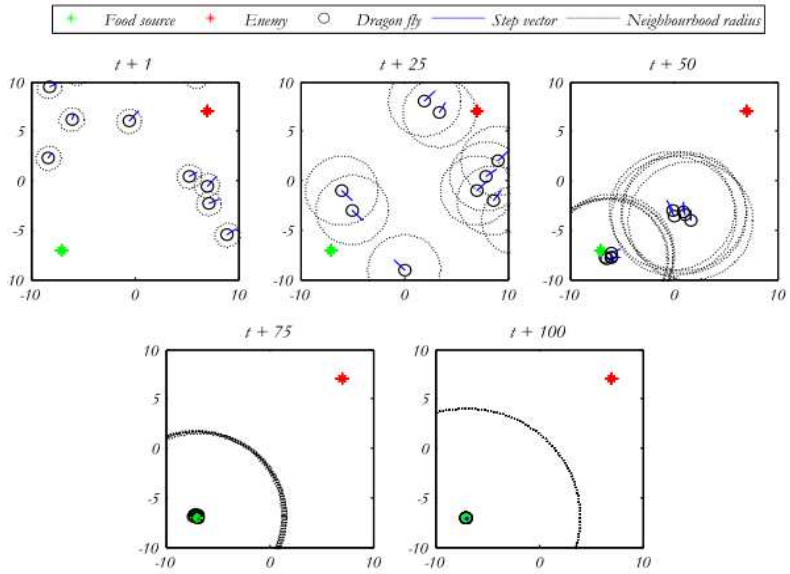


Fig 5.7 Swarming behaviour of artificial dragonflies

The pseudo-code of the DA is:

Initialize the dragonflies population X_i (i = 1, 2, ..., n)
Initialize step vectors ΔX_i (i = 1, 2, ..., n)
***while** the end condition is not satisfied*
    Calculate the objective values of all dragonflies
    Update the food source and enemy
    Update w, s, a, c, f, and e
    Calculate S, A, C, F, and E using Eqs. (3.1) to (3.5)
    Update neighbouring radius
    ***if** a dragonfly has at least one neighbouring dragonfly*
        Update velocity vector using Eq. (3.6)
        Update position vector using Eq. (3.7)
    else
        Update position vector using Eq. (3.8)
    end if
    Check and correct the new positions based on the boundaries of variables
end while

Fig. 5.8 pseudo-code of DA

6.Simulation and results

6.1 Fault implementation

To increase the convergence speed of the optimization the monitor model is simulated with normalized faults parameters which values are limited in a interval [0,1]. A vector of faults is formed with these parameters:

$$k=[k(1),k(2),k(3),k(4),k(5),k(6),k(7),k(8)]$$

where:

- k(1) is the normalized friction fault, at 0 it represents the nominal condition and at 1 it is three time the nominal condition.

- k(2) is the normalized backlash fault, at 0 it represents the nominal condition equal to a backlash of 0.29° and at 1 it is ten time the nominal condition, 29°.
- k(3 to 5) are the shot circuit parameters that range form a nominal condition 0 that is equal to the 100% current passing through the phases A, B or C, to 1 which represent the short circuit.
- k(6) the normalized rotor eccentricity ratio, ranges from the nominal condition that represents the ideal absence of static eccentricity , to 1 which is equal to g=0 (no air gap between rotor and stator)
- k(7) is the second parameters related to the eccentricity, it is the phase of eccentricity normalized with 0=-π to 1= π, while the nominal condition is set at 0.5. This parameters is not considered when k(6)=0.
- K(8) is the normalized gain fault, it's nimal condition are set at 0.5, while 0 represents a reduction 50% and 1 is equal to an increase of 50% in gain.

In nominal condition the vector of faults is eqaul to:

$$k = [0{,}0\,,0{,}0\,,0{,}0\,,0{,}0\,,0{,}0\,,0.5\,,0.5]$$

After the simulation of the reference model, the faulty parameters are normalized and then the monitor model is launched with the current vector of fault and compute the error between the equivalent currents of both models with a dedicated fitness function, that must be minimized. All this processes is iterated multiple times during one optimization, the algorithm will try to find the best vector of faults between an initial random pool of solution while also improving this pool. For this reason the selection of a proper fitness function is crucial.

6.2 Fitness function

A proper fitness function has been used for the previous algorithms implemented on the software, it is obtained with the total least square method in order detect the smallest difference between the two models:

$$err = \frac{\sum \left(I_{3\text{equiv}}(t_0) - I_m(t_0)\right)^2}{\sqrt{\left(\frac{\left(I_{3\text{equiv}}(t_0)^2\right)}{dt} + 1\right)}}$$

where I_{3equiv} and I_m are the reference equivalent current and the monitor current.

Some simulations without the optimization algorithm are showed below in order to check the reliability of the fitness function. For each fault there are three different results, the first for nominal condition, the second for a low faulty condition and the third for a high faulty condition.

Good results should show a very low error.

FRICTION

F=1 k(1)=0 $err = 3.9 \cdot 10^{-5}$

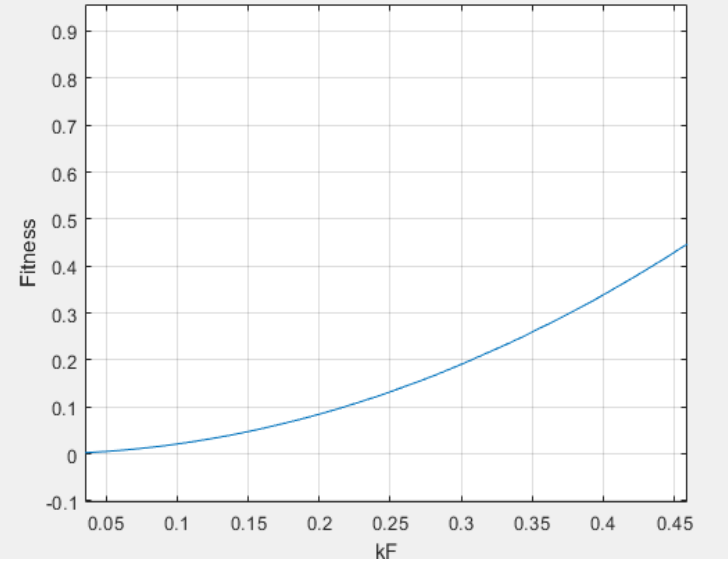


Fig 6.1 Nominal friction

F=1.5 k(1)= 0.25 $err = 8.7 \cdot 10^{-4}$

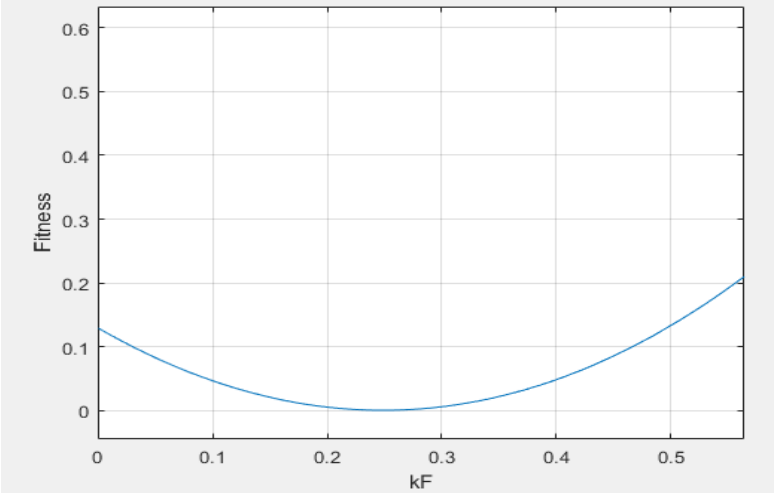


Fig 6.2 Low friction

F= 2.5 k(1)= 0.75 $err = 2.9 \cdot 10^{-3}$

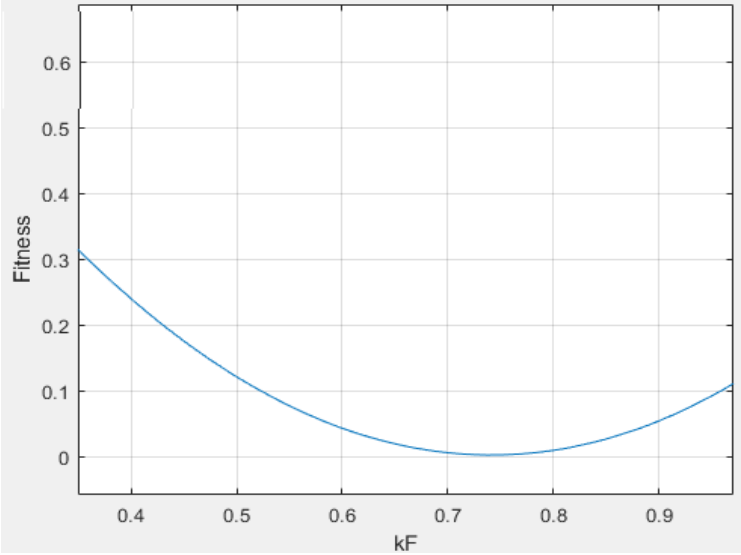


Fig 6.3 High friction

Nominal conditions are the bests, the errors increase of 10 times for the low faulty case and 100 times for the high faulty one. The results are acceptable.

B=0 k(2)=0 $err = 3.9 \cdot 10^{-5}$

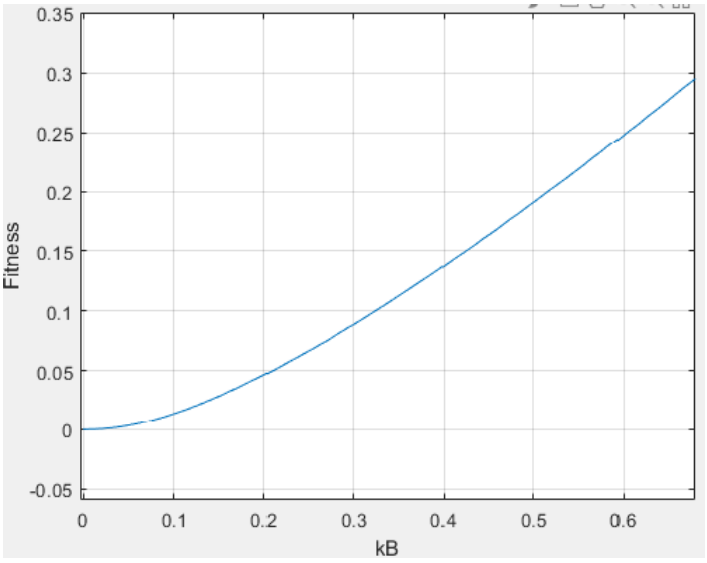


Fig 6.4 Nominal Backlash

B=25 k(2)=0.25 $err = 2.2\ 10^{-4}$

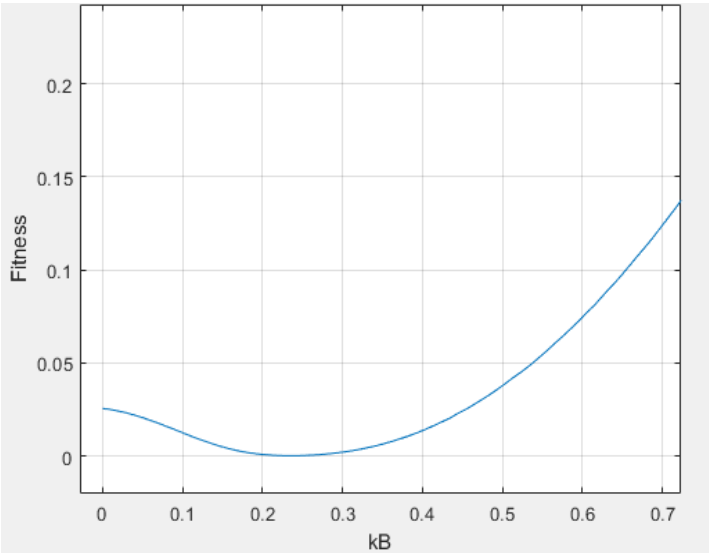


Fig. 6.5 Low backlash

B=75 k(2)=0.75 $err = 2.2\ 10^{-4}$

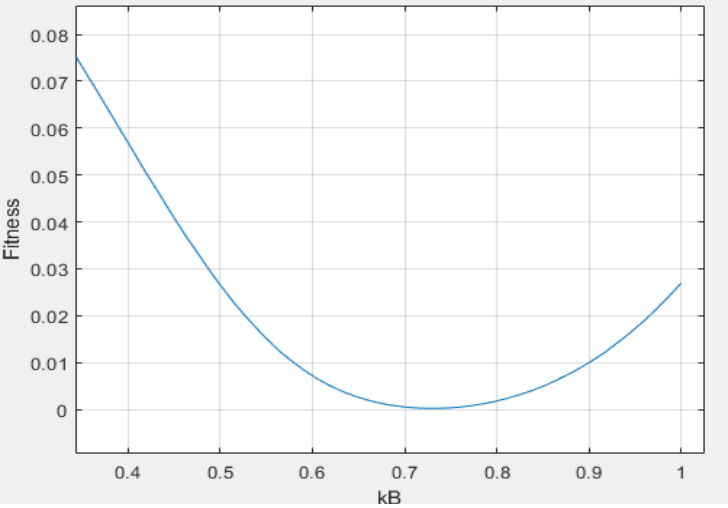


Fig. 6.6 high backlash

For the backlash the final results are also good, as always there is a lack of precision between nominal condition and both high and low fault that shows an error ten times greater.

SHORT CIRCUIT

Na=1 k(3)=0 $err = 4.8\ 10^{-4}$

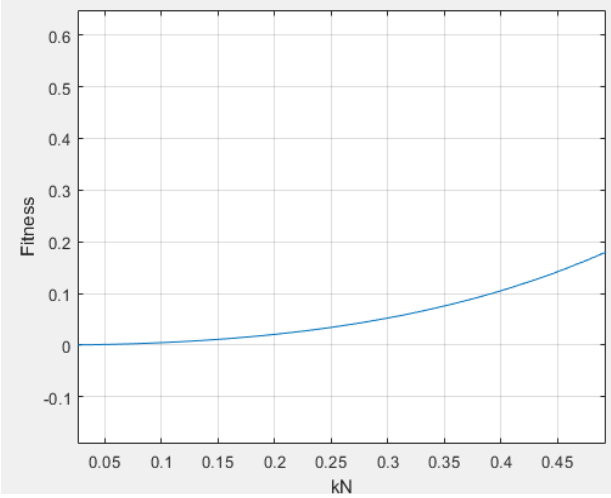


Fig 6.7 Nominal shot circuit

Na=0.8 k(3)=0.2 $err = 6.8\ 10^{-4}$

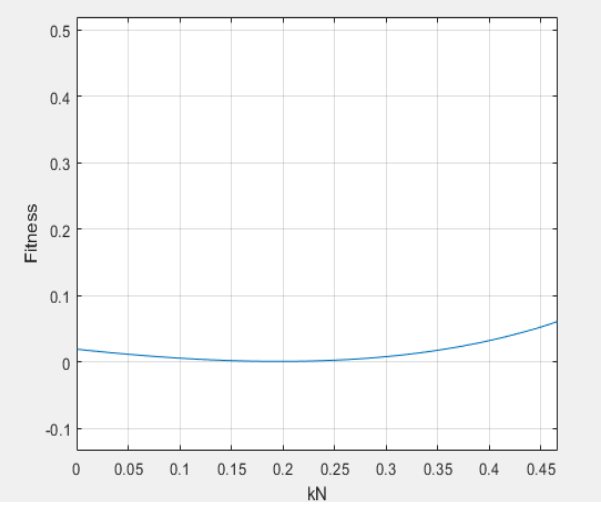


Fig 6.8 Low short circuit

Na=0.3 k(3)=0.7 $err = 2.1 \cdot 10^{-3}$

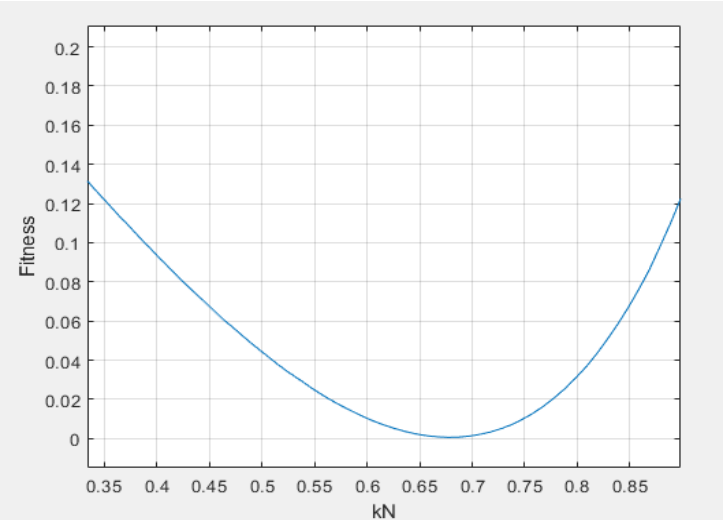


Fig 6.9 High short circuit

The error of the low fault case is just two time greater then the nominal one, while the high fault is ten times but still acceptable.

GAIN

G=1 k(8)=0.5 $err = 6.2 \cdot 10^{-5}$

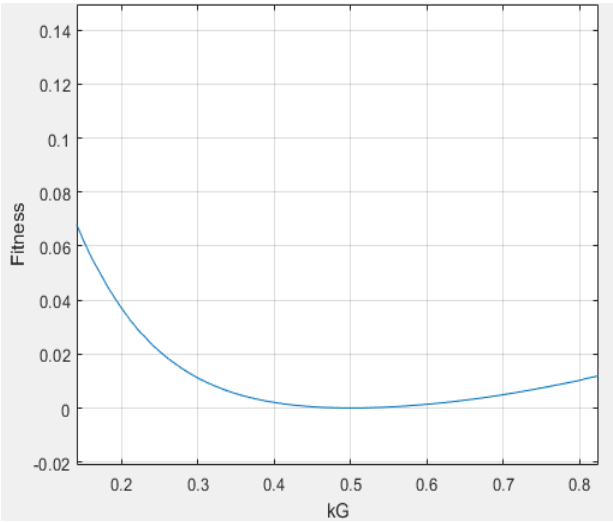


Fig. 6.10 Nominal gain

G=0.5 k(8)=0.25 $err = 7.9 \cdot 10^{-2}$

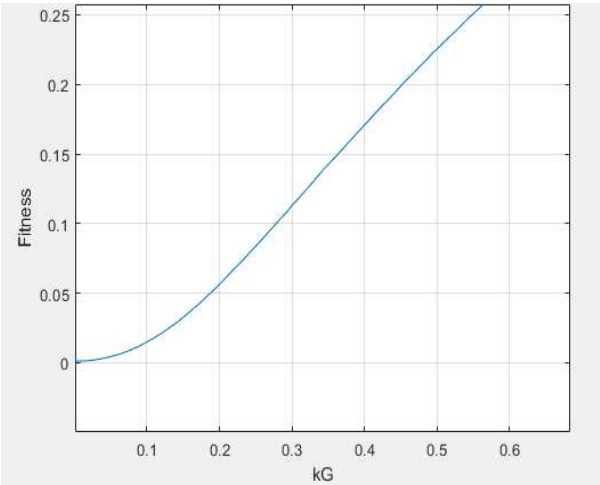


Fig. 6.11 Low gain

G=1.5 k(8)=0.75 $err = 3.4 \cdot 10^{-3}$

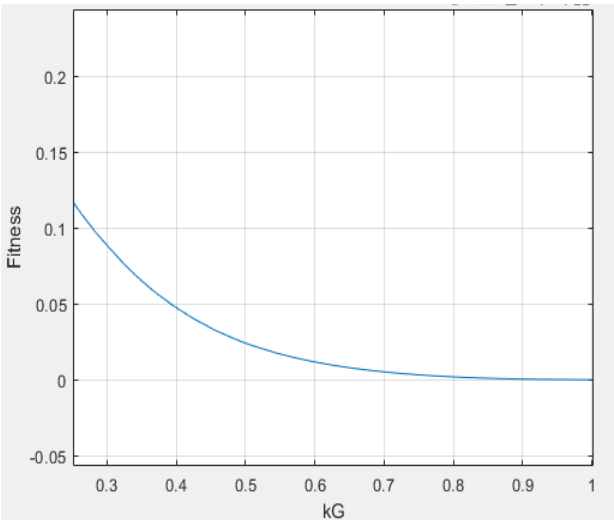


Fig. 6.12 high gain

In this case, the low fault generates a greater error than anything considered before, therefore I could be an issue for the simulation, while the others are similar to the other faults.

ECCENTRICITY

Z=0. k(6)=0 $err = 3.9 \cdot 10^{-5}$

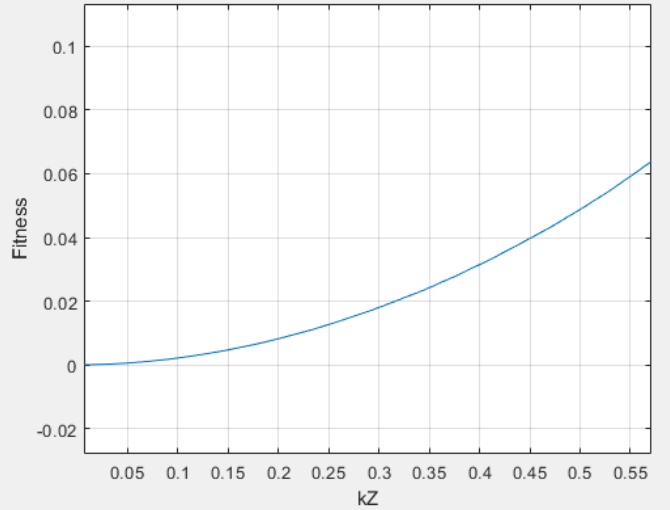


Fig. 6.13 Nominal eccentricity ratio

Z=0.105 k(6)=0.25 $err = 2.7 \cdot 10^{-4}$

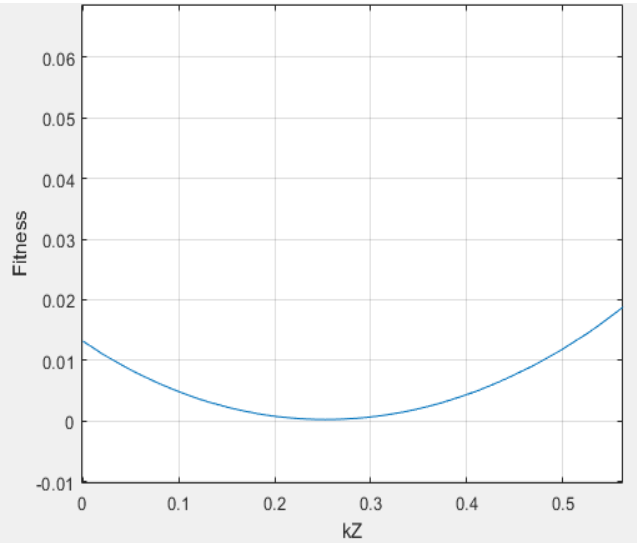


Fig 6.14 Low eccentricity ratio

Z=0.315 k(6)=0.75 $err = 1.9 \cdot 10^{-2}$

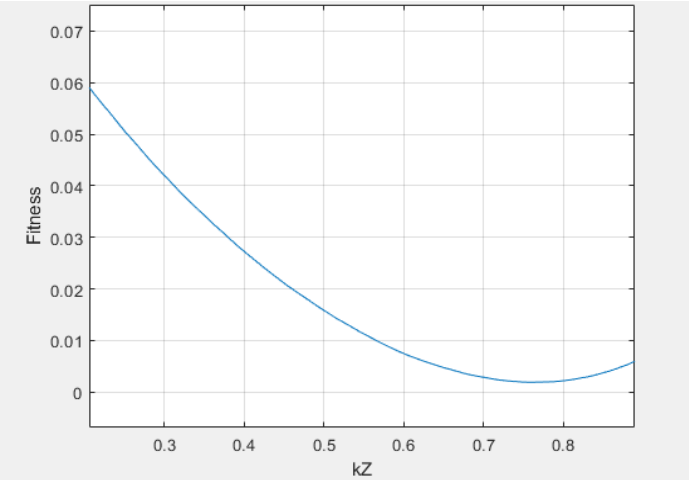


Fig. 6.15 High eccentricity ratio

For the eccentricity ratio, the high faulty case generates a big error while the others shows good results.

phi=0° k(7)=0.5 $err = 2.6\ 10^{-4}$

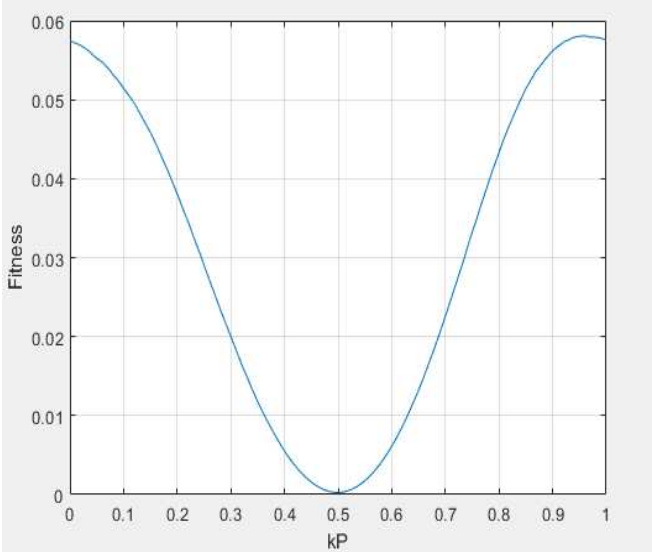


Fig. 6.16 nominal eccentricity phase

phi=-180° k(7)=0.25 $err = 7.1 \cdot 10^{-4}$

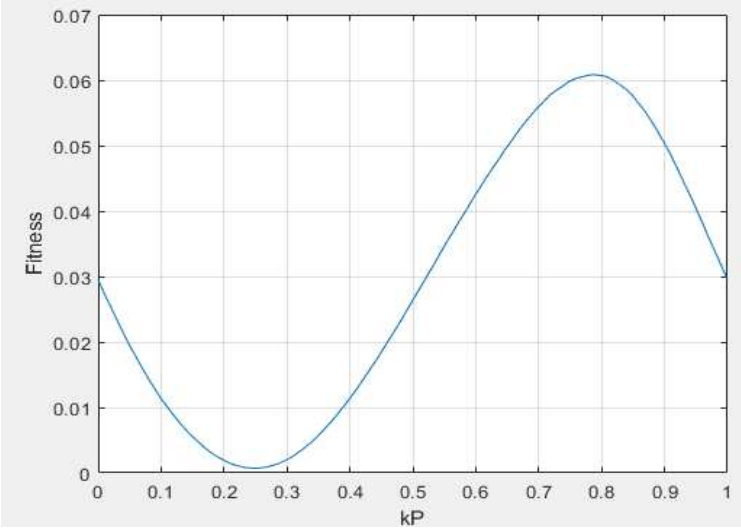


Fig. 6.17 Low eccentricity phase

phi=+180° k(7)=0.75 $err = 8.1 \cdot 10^{-4}$

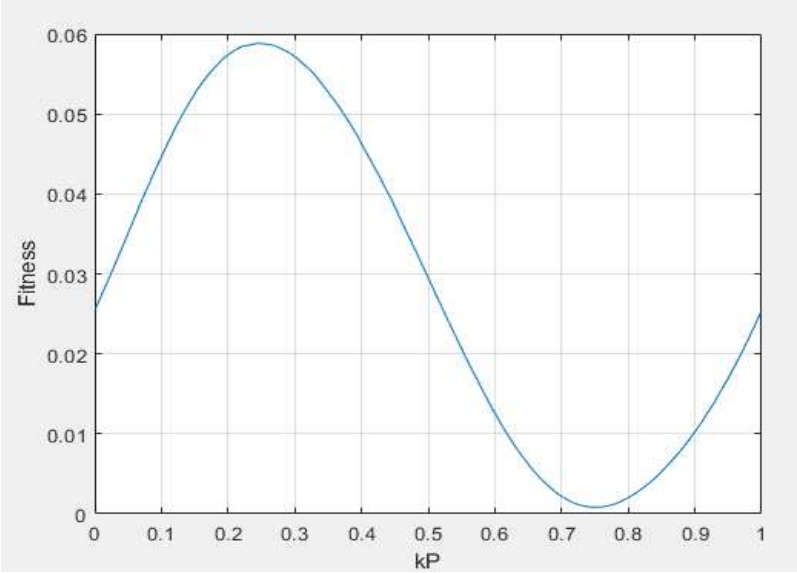


Fig. 6.18 High eccentricity phase

The phases of the eccentricity, which is simulated with a k(6)=0.105, shows good results for all the faults with each errors belonging to the same order.

6.3 Parameters for the optimization

Here is a list for the parameters that can be selected for the optimization:

- Population number/number of search agents: is the number of individual in a population/swarm, it represents the number of possible solution, each one of them is a possible vector of fault. The higher is this number, the better is the precision but also the expansive is the computational cost.
- Iterations/generation: is the number that regulates how many times the operation of the algorithm are executed. Like the number of search agent, it should be chosen considering that after a certain threshold the precision increment is little, while the computation time is too high.
- Parallelization: it can be set ON or OFF, it's a tool that enable the simulation of the model on multiple cores of the processor. It is always set OFF.
- MyC parameter: this is a special parameter for the dragonflies algorithm, it consist in a general weight parameters that affect all the values obtained by the primitive corrective patterns equation. After different trials, it is defined as:
- $$MyC = 0.01 - \frac{(0.02 \cdot iter)}{MaxIter}$$

where iter is the number of the current iteration and MaxIter is the number of the last iteration.

6.4 Performance

To evaluate the algorithm two coefficients are used:

- Error: this equation takes in account of every fault error and computes a total error:

$$Err = \sqrt{\left(\sum_{i=1}^{5} (K_i - k_i)^2 + k_6 (K_7 - k_7) + (K_8 - k_8)^2\right)}$$

where K are the nominal values and k are the faulty ones.

As the equation shows, the error of the eccentricity is equal to 0 if the ratio is in nominal condition, that is to avoid a physically meaningless results if just the phase fault is present.

- Reliability coefficient: this parameters takes compute a reliability value to have an indicator of the efficiency of the utilized algorithm, so it can be compared to the other optimization methods. A 100% RC represent the best method possible.

$$RC = 1 - \frac{\left(t_i \, err_i\right)}{N} \Big/ \left(\sum_{i=1} t_i \, err_i\right)$$

- where err_i is the mean of the total errors of all the optimizations carried by the i-esim algorithm, N is the number of algorithms, t_i is the mean computational time of the i-esim algorithm.

6.5 Single fault detection

The simulations were carried with a number of search agents equal to 50 and a number of iterations equal to 100. Both numbers were selected after many trials, it is important to know that increasing both parameters will only generates an higher computational time, while the precision does not change in a significant way.

For each fault one high faulty case and one low faulty case are simulated, in the table below the vector of faults used is listed. Also, every case has been executed for a total of 1o optimizations.

F-k(1)	0,25	0	0	0	0	0	0,5	0,5
B-k(2)	0	0,25	0	0	0	0	0,5	0,5
Na-k(3)	0	0	0,2	0	0	0	0,5	0,5
Z-k(6)	0	0	0	0	0	0,25	0,5	0,5
G-k(8)	0	0	0	0	0	0	0,5	0,25

Low faults vectors

F-k(1)	0,75	0	0	0	0	0	0,5	0,5
B-k(2)	0	0,75	0	0	0	0	0,5	0,5
Na-k(3)	0	0	0,8	0	0	0	0,5	0,5
Z-k(6)	0	0	0	0	0	0,75	0,5	0,5
G-k(8)	0	0	0	0	0	0	0,5	0,75

High fault vectors

Friction

Optimization	**Low F**								**error**	**time**
1	0,2479	0	0	0	0	0	0	0,4912	0,9	946
2	0,2482	0	0	0	0	0	0,5938	0,4875	1,26	952
3	0,2481	0	0	0	0	0	0,1748	0,4911	0,9	1.014
4	0,2489	0	0	0	0	0	1	0,4950	0,88	1.039
5	0,2485	0,0019	0	0	0	0	1	0,4912	0,91	1.025
6	0,2484	0	0	0	0	0	0	0,4826	1,74	1.005
7	0,2479	0	0	0	0	0	0,6158	0,4902	1,01	1.019
8	0,2372	0	0	0	0	0	0,1938	0,4808	2,3	1.026
9	0,2480	0	0	0	0	0	0,0219	0,4908	0,92	1.069
10	0,2488	0	0	0	0	0	0	0,4941	0,61	1.078
avg error	1,14	**avg time**	1.017							

Low F fault

Optimization	High F									**time**
1	0,8006	0	0	0	0	0	0	0,4786	5,49	915
2	0,7378	0	0	0	0	0,0024	0	0,4758	2,72	932
3	0,7401	0	0	0	0	0	1	0,4776	2,44	948
4	0,7394	0	0	0	0	0	0	0,4827	1,99	963
5	0,7351	0	0	0	0	0	0,3782	0,4776	2,32	979
6	0,7391	0	0	0	0	0	0	0,4786	1,98	997
7	0,7377	0	0	0	0	0	1	0,4789	1,99	1.010
8	0,7390	0	0	0	0	0	0	0,4787	1,97	1.027
9	0,8006	0	0	0	0	0	0,4091	0,4767	5,37	1.043
10	0,7388	0	0	0	0	0	1	0,4782	2,66	1.095
avg error	2,89	**avg time**	991							

High fault

Both faults shows decent performances, the high fault error is three time greater then the low fault but that was predictable since the fitness function behaves better at low levels of friction.

Backlash

Optimization	low B									**time**
1	0,0376549824	0,2371	0	0	0	0	0,9135	0,5113	1,8	910
2	0	0,2355	0	0	0	0	0,4092	0,5010	1,45	917
3	0	0,2356	0	0	0	0	0,0635	0,5017	1,51	913
4	0	0,2374	0	0	0	0	0	0,4978	1,28	922
5	0	0,2365	0	0	0	0	0,5917	0,4978	1,37	917
6	0	0,2379	0	0	0	0	0	0,5747	7,5	922
7	0	0,2364	0	0	0	0	0,6667	0,4978	1,59	1.001
8	0	0,2368	0	0	0	0,1823	0,5029	0,5788	7,9	1.036
9	0	0,2494	0	0,0138	0	0,0166	0,0427	0,5013	1,24	1.049
10	0	0,2433	0	0,0000	0	0	0	0,5747	7,5	1.062
avg error	3,31	**avg time**	964							

Optimization	high B									**time**
1	0	0,7323	0	0	0	0	0,6095	0,4974	1,78	971
2	0	0,7325	0	0	0	0	0	0,4979	1,75	975
3	0	0,7325	0	0	0	0	0,5740	0,4979	1,75	949
4	0	1	0	0	0	0	0,1054	0,7467	35,11	992

5	0	0,7325	0	0	0	0	0,3563	0,4982	1,75	1.030
6	0	0,7155	0	0	0	0	0,4823	0,4969	3,46	1.017
7	0	1	0	0	0	0	0	0,7467	35,11	1.037
8	0	0,7325	0	0	0	0,2000856545	1	0,5289	44,81	1.061
9	0	0,7339	0	0	0	0	0,8939	0,4972	1,63	1.098
10	0	0,6448	0	0	0	0,2426568175	0,5092	0,4628	11,16	1.153
avg error	14	**avg time**	1.027							

The backlash is well detected for the low fault, while the high fault has some problems of convergence, for multiple optimization the algorithm diverge to the upper boundary.

Short circuit (Na)

Optimization	low Na									**time**
1	0	0	0,1970	0	0	0	1	0,4991	0,32	1.108
2	0	0	0,1963	0	0	0	1	0,4971	0,47	1.038
3	0	0	0	0	0	0,1177	0	0,5423	20,44	974
4	0	0	0,1935	0	0	0	0,1915	0,5016	0,67	1.006
5	0	0	0,1657	0	0	0,0066	0,8542	0,4959	3,81	1.021
6	0	0	0,1428	0	0	0,0176	0,2410	1,0000	50,3	1.028
7	0	0	0	0	0	0,1483	1	0,5023	20,32	1.038
8	0	0	0,1937	0	0	0	0,0640	0,5014	0,71	1.062

9	0	0	0,1974	0	0	0	0	0,4979	0,34	1.074
10	0	0	0	0	0	0,1695	1	0,4569	31,45	1.086
avg error	12,83	**avg time**	953							

Optimization	high Na									**time**
1	0	0	0,6900	0,0693	0,0936	0	0	0,5117	18,13	986
2	0	0	0,6643	0	0,1847	0,2097	0,4746	0,5721	42,6	999
3	0	0	0,7811	0	0	0	0	0,4961	1,92	1.007
4	0	0	0,6306	0,1825	0,0456	0	0	0,5728	26,34	992
5	0	0	0,7780	0	0,0008	0	0,8017	0,5158	2,71	950
6	0	0	0,7786	0	0	0	1	0,5078	2,25	947
7	0	0	0,6793	0	0,0473	0,1607	1	0,3691	24,25	1.024
8	0	0	0,7753	0	0	0	0,7103	0,5225	3,34	1.064
9	0	0	0,7366	0	0	0,1412	1	0,4796	38,1	1.080
10	0	0	0,7792	0	0	0	1	0,5154	2,58	1.095
avg error	16,22	**avg time**	1.014							

The short circuit has the worst results so far, having hard times at detecting both fault.

Eccentricity

Optimization	low K									**time**
1	0	0	0	0	0	0	0,3967	0,4614	50,1	1.020
2	0	0	0	0	0	0	0	0,4486	50,26	1.099
3	0	0,0314	0	0	0	0	0,0946	0,4384	50,37	1.069
4	0	0	0	0	0	0	0,0786	0,4614	50	1.151
5	0	0	0	0	0	0,2494	0,4983	0,5009	0,1	1.077
6	0	0	0	0	0	0	0	0,4493	50,61	998
7	0	0	0	0	0	0	0,1563	0,4487	50,63	1.047
8	0	0	0	0	0	0,2544	0,4988	0,5031	3,35	1.060
9	0	0,0297	0	0	0	0	0,4829	0,4406	50,77	1.060
10	0	0	0	0	0	0,2571	0,4993	0,5012	0,71	1.077
avg error	35,63	**avg time**	1.065							

Optimization	high K									**time**
1	0	0	0	0	0	1	0	0,3695	13,05	951
2	0	0	0	0	0	0,5228	0,4993	0,5328	16,34	922
3	0	0	0	0	0	0,7652	0,4996	0,4991	8,71	918
4	0	0	0	0	0	1	0	0,3182	18,26	924
5	0	0	0	0	1	0,5166	1	0,5454	48,52	931
6	0	0	0	0	0	0,7606	0,5012	0,4978	7,3	981

7	0	0	0	0	0	0,7518	0,4998	0,4884	3,21	1.019
8	0	0	0	0	0	0,7426	0,5045	0,4802	6,41	1.034
9	0	0	0	0	0	0,5282	0,4991	0,5142	33,3	1.039
10	0	0	0	0	0	0	0	0,3358	16,42	1.063
avg error	19,23	**avg time**	978							

The algorithm seems to be not suited for the eccentricity fault detection, the errors are the worst of every fault and both ratio and phases are poorly detced.

Gain

Optimization	low G									939
1	0	0	0	0	0	0	0,9402	0,2549	0,49	925
2	0	0	0	0	0	0	0,1417	0,2549	0,49	980
3	0	0	0	0	0	0	0,1982	0,2549	0,49	961
4	0	0	0	0	0	0	0	0,2549	0,49	965
5	0	0	0	0	0	0	0	0,2264	2,36	976
6	0	0	0	0	0	0	0	0,2549	0,49	946
7	0	0	0	0	0	0	0,4185	0,2549	0,49	926
8	0	0	0	0	0	0	0,0655	0,2549	0,49	930
9	0	0	0	0	0	0	0,2737	0,2549	0,49	940
10	0	0	0	0	0	0	0	0,2549	0,49	949
avg error	0,67	**avg time**	949							

Optimization	high G									**time**
1	0	0	0	0	0	0	0	0,7342	1,58	1.059
2	0	0	0	0	0	0	0	0,7342	1,58	1.115
3	0	0	0	0	0	0	1	0,7337	1,56	972
4	0	0	0	0	0	0	0,2829	0,7342	1,58	917
5	0	0	0	0	0	0	0	0,7342	1,58	918
6	0	0	0	0	0	0	0	0,7342	1,58	918
7	0	0	0	0	0	0	0	0,7334	1,66	920
8	0	0	0	0	0	0	0,7584	0,7342	1,58	922
9	0	0	0	0	0	0	0,2185	0,7344	1,58	915
10	0	0	0	0	0	0	0	0,7342	1,58	1.009
avg error	1,58	**avg time**	966							

The Gain is by far the best, both errors are good even if the fitness function does not behave at the best for this fault.

6.6 Multiple fault detection

As anticipated from the single fault detection, the results of the dragonflies algorithm applied to the current model are not satisfied. Below there are the optimizations for the multiple fault detection.

Optimization	0,25	0,75	0,8	0,8	0,8	0,25	0,5	0,75	**error**	**time**
1	0,4678	1	0,7516	0,5953	0,8509	0,8654	0,5766	0,7928	71,6	1.026
2	0,7035	0,8320	0,7036	0,7533	0,7434	0,0026	0,6751	1	67,57	1.039
3	0,6941	0,7875	0,7046	0,6851	0,8091	0,1436	0,7965	1	60,69	1.040
4	0,6945	0,7226	1	0,4742	0,5963	0,6032	1	0,6128	80,77	1.051
5	0,6002	0,7495	0,7526	0,7020	0,7638	0,7799	1	0,6164	79,91	1.056
6	0,2716	0,7534	0,7291	0,7309	0,8745	0,6343	1	0,6267	48,77	1.068
7	0,3076	1	0,7892	0,6175	0,8217	0,8698	0,5506	0,9496	74,6	1.074
8	0,2722	0,2529	0,8472	0,7378	0,7384	0,0043	0,8378	0,5221	66,5	1.095
9	0,5168	1	0,7357	0,7304	0,7553	0,2150	0,5318	1	47,48	1.095
10	0,4949	0,5137	0,7881	0,6689	0,8016	0,1564	0,4941	0,5821	45,56	1.102
avg error	64,2	**avg time**	1064							

The average error shows that the current algorithm is not suited for this application.

6.7 Comparison and conclusion

Some results has been taken from previous works in order to compare the DA and achieve a better vision of its performance.

	GA	PSO	DE	GWO	DA
low F	2,50%	1,06%	1,11%	1,19%	1,14%
high F	4,08%	2,82%	3,41%	2,58%	2,89%
low B	1,96%	1,26%	1,26%	1,27%	3,31%
high B	2,08%	2,01%	1,94%	1,79%	14,00%
low Na	1,66%	0,44%	0,42%	0,78%	12,83%
high Na	8,06%	3,40%	3,11%	3,18%	16,22%
low K	1,24%	3,88%	0,38%	1,49%	35,63%
high K	1,60%	1,52%	1,71%	4,61%	19,23%
low G	1,63%	0,41%	0,43%	0,35%	0,67%
high G	1,27%	0,19%	0,27%	0,27%	1,58%
Average	2,57%	1,70%	1,40%	1,75%	10,75%

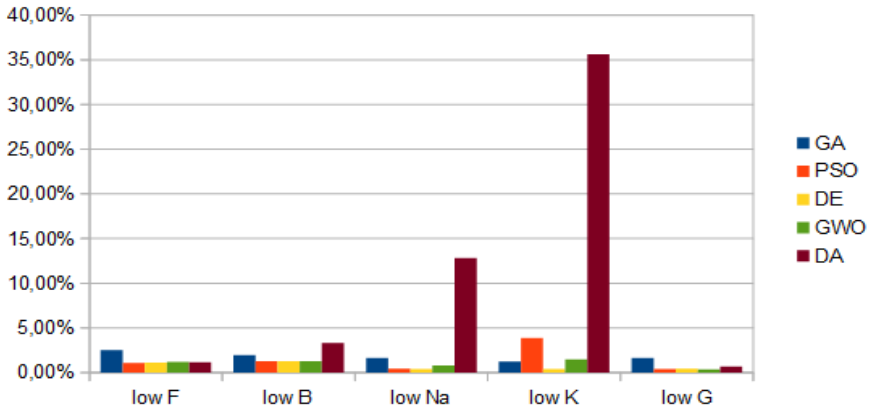


Fig. 6.19 low faults errors comparison

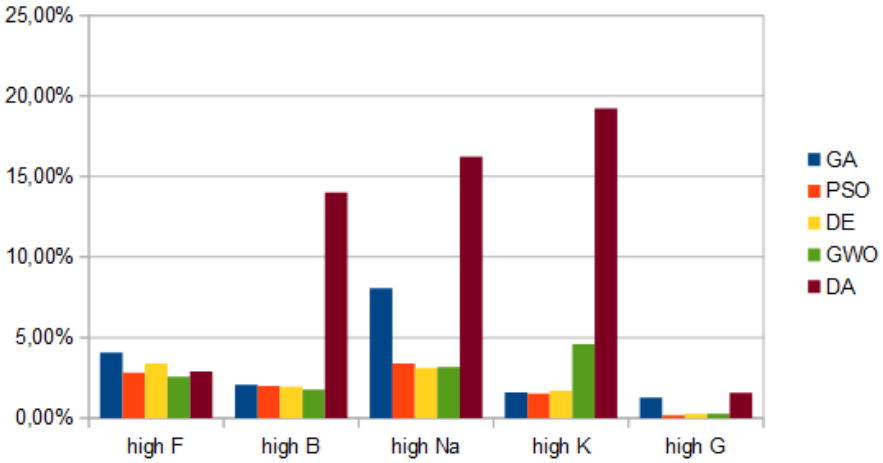
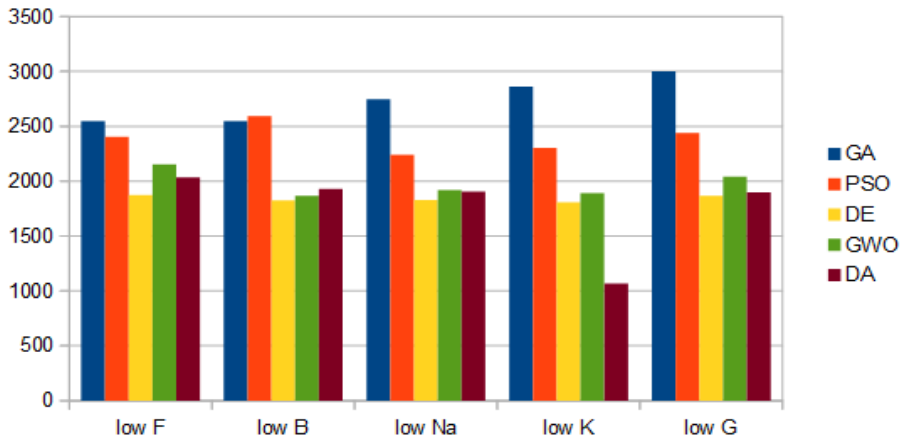


Fig. 6.20 high faults errors comparison

The Dragonflies algorithms shows competitive results on friction and gain detection, while the rest for the rest of faults the performance are not satisfactory.

	GA	PSO	DE	GWO	DA

low F	2549	2405	1874	2152	2.034
high F	2764	2589	1717	1893	1982
low B	2549	2593	1824	1867	1.928
high B	2737	2428	1809	1955	2.054
low Na	2745	2240	1829	1918	1.904
high Na	2882	2317	1859	1983	2.028
low K	2862	2304	1806	1892	1.065
high K	2735	1933	1739	1966	1956
low G	3002	2438	1867	2043	1.898
high G	3032	1873	1922	1957	1.928
Average	2786	2312	1825	1963	1.984



6.21 low faults time comparison

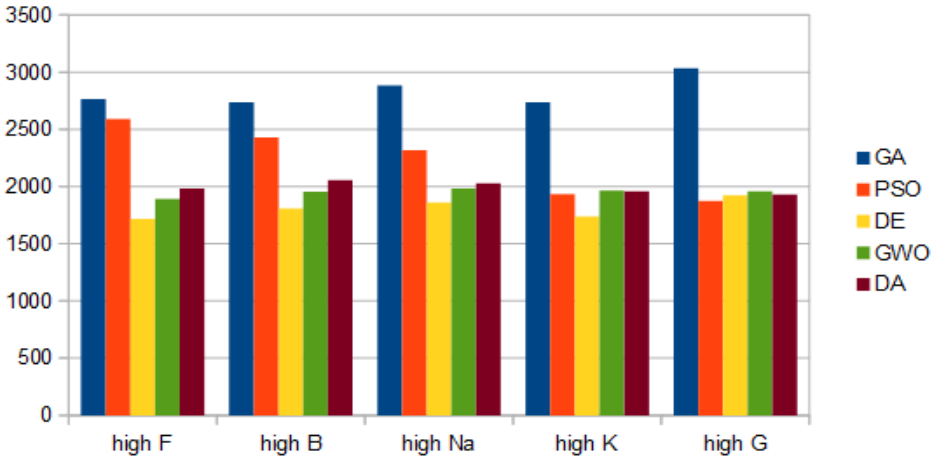


Fig. 6.21 high fault time comparison

For the time comparison the results of dragonflies are multiplied by 2, since it was used with half of the iterations. In this case it just performed like the most parts of the algorithms.

	GA	PSO	DE	GWO	DA
Time	2786	2312	1825	1963	1.984
Err	2,57%	1,70%	1,40%	1,75%	10,75%
RC	76,00%	91,00%	93,00%	92,00%	62,00%

In the end, the DA does have the lowest reliability coefficient among the algorithms studied, it is not suited for this application, while PSO, DE and GWO are the best choice for carrying out the optimization of this EMA model.

In the future, an hybrid algorithm will probably be the best way to achieve this task, since all the presented methods have their weakness. In this sense, a note of praise must be done for the DA, since its computational time is the lowest seen here, and that's an important feature for on-board fault detection.

Appendix

```
function
[Best_score,Best_pos,cg_curve]=DA(SearchAgents_no,Max_iteration,lb,ub,dim,fob
j)

 display('DA is optimizing your problem');

cg_curve=zeros(1,Max_iteration);

 if size(ub,2)==1

    ub=ones(1,dim)*ub;

    lb=ones(1,dim)*lb;

end

 %The initial radius of gragonflies' neighbourhoods

r=(ub-lb)/10;

Delta_max=(ub-lb)/10;

Food_fitness=inf;

Food_pos=zeros(dim,1);


Enemy_fitness=-inf;

Enemy_pos=zeros(dim,1);


X=initialization(SearchAgents_no,dim,ub,lb);

Fitness=zeros(1,SearchAgents_no);


DeltaX=initialization(SearchAgents_no,dim,ub,lb);


for iter=1:Max_iteration
```

```
  r=((ub-lb)/4)+((ub-lb)*((iter/Max_iteration)*2));

  w=0.9-iter*((0.9-0.4)/Max_iteration);
%c / 10
     my_c=0.01-iter*((0.01-0)/(Max_iteration/2));
  if my_c<0
      my_c=0;
  end

  s=2*rand*my_c; % Seperation weight
  a=2*rand*my_c; % Alignment weight
  c=2*rand*my_c; % Cohesion weight
  f=2*rand;      % Food attraction weight
  e=my_c;        % Enemy distraction weight




  %Calculate all the objective values first
  for i=1:SearchAgents_no %Calculate all the objective values first

      Fitness(1,i)=fobj(X(:,i)');
      All_fitness(1,i)=Fitness(1,i);
      if Fitness(1,i)<Food_fitness
          Food_fitness=Fitness(1,i);
          Food_pos=X(:,i);
      end
              if Fitness(1,i)>Enemy_fitness
          if all(X(:,i)<ub') && all( X(:,i)>lb')
              Enemy_fitness=Fitness(1,i);
              Enemy_pos=X(:,i);
```

```
        end
    end
end

for i=1:SearchAgents_no
    index=0;
    neighbours_no=0;

    clear Neighbours_V
    clear Neighbours_DeltaX
    %find the neighbouring solutions
    for j=1:SearchAgents_no
        Dist2Enemy=distance(X(:,i),X(:,j));
        if (all(Dist2Enemy<=r) && all(Dist2Enemy~=0))
            index=index+1;
            neighbours_no=neighbours_no+1;
            Neighbours_DeltaX(:,index)=DeltaX(:,j);
            Neighbours_X(:,index)=X(:,j);
        end
    end

    % Seperation%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Eq. (3.1)
    S=zeros(dim,1);
    if neighbours_no>1
        for k=1:neighbours_no
            S=S+(Neighbours_X(:,k)-X(:,i));
        end
        S=-S;
    else
        S=zeros(dim,1);
```

```
        end


% Alignment%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Eq. (3.2)
        if neighbours_no>1
            A=(sum(Neighbours_DeltaX')')/neighbours_no;
        else
            A=DeltaX(:,i);
        end

        % Cohesion%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Eq. (3.3)
        if neighbours_no>1
            C_temp=(sum(Neighbours_X')')/neighbours_no;
        else
            C_temp=X(:,i);
        end

        C=C_temp-X(:,i);

        % Attraction to food%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Eq. (3.4)
        Dist2Food=distance(X(:,i),Food_pos(:,1));
        if all(Dist2Food<=r)
            F=Food_pos-X(:,i);
        else
            F=0;
        end
```

```
        % Distraction from enemy%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Eq. (3.5)
        Dist2Enemy=distance(X(:,i),Enemy_pos(:,1));
        if all(Dist2Enemy<=r)
            Enemy=Enemy_pos+X(:,i);

 else
            Enemy=zeros(dim,1);
        end

        for tt=1:dim
         if X(tt,i)>ub(tt)
           X(tt,i)=lb(tt);
            DeltaX(tt,i)=rand;
                end
         if X(tt,i)<lb(tt)
            X(tt,i)=ub(tt);
           DeltaX(tt,i)=rand;
                end
            end




        if any(Dist2Food>r)
            if neighbours_no>1
                for j=1:dim
                    DeltaX(j,i)=w*DeltaX(j,i)
+rand*A(j,1)+rand*C(j,1)+rand*S(j,1);
                    if DeltaX(j,i)>Delta_max(j)
```

```
                        DeltaX(j,i)=Delta_max(j);
                    end
                    if DeltaX(j,i)<-Delta_max(j)
                        DeltaX(j,i)=-Delta_max(j);
                    end
                    X(j,i)=X(j,i)+DeltaX(j,i);

 end
            else
                % Eq. (3.8)
                X(:,i)=X(:,i)+Levy(dim)'.*X(:,i);
                DeltaX(:,i)=0;
            end
        else
            for j=1:dim
                % Eq. (3.6)

DeltaX(j,i)=(a*A(j,1)+c*C(j,1)+s*S(j,1)+f*F(j,1)+e*Enemy(j,1)) +
w*DeltaX(j,i);
                if DeltaX(j,i)>Delta_max(j)
                    DeltaX(j,i)=Delta_max(j);
                end
                if DeltaX(j,i)<-Delta_max(j)
                    DeltaX(j,i)=-Delta_max(j);
                end
                X(j,i)=X(j,i)+DeltaX(j,i);
            end
        end
        Flag4ub=X(:,i)>ub';
        Flag4lb=X(:,i)<lb';
        X(:,i)=(X(:,i).*(~(Flag4ub+Flag4lb)))+ub'.*Flag4ub+lb'.*Flag4lb;
```

```
    end


    Best_score=Food_fitness;
    Best_pos=Food_pos;
    cg_curve(iter)=Best_score;

     if iter>2
        line([iter-1 iter], [cg_curve(iter-1) cg_curve(iter)],'Color','b')
        xlabel('Iteration');
        ylabel('Best score obtained so far');
        drawnow
    end


        hold on
        scatter(iter*ones(1,SearchAgents_no),All_fitness,'.','k')
        criterium=2e-5;

end

end
```