MUNER – Motorvehicle University of Emilia Romagna

ADVANCED AUTOMOTIVE ELECTRONICS ENGINEERING

PROJECT REPORT

Compliance Design of Automotive Systems

BLAC Motor Control

Giorgia Falbo

Edoardo Gardani

A. A. 2018/2019

27th March 2019

Summary

1 INTRODUCTION

The aim of this project is to exploit some simulation and automatic tool in order to define the control parameter of PMSM and to generate the code for a C environment using a Model-Based Design.

Starting from a predefined system model made in Simulink and Simscape of a Permanent Magnet Synchronous Machine (PMSM) and an inverter sized for use in a typical hybrid vehicle, we modelled some non-ideal behaviour of the system and we applied an external torque.

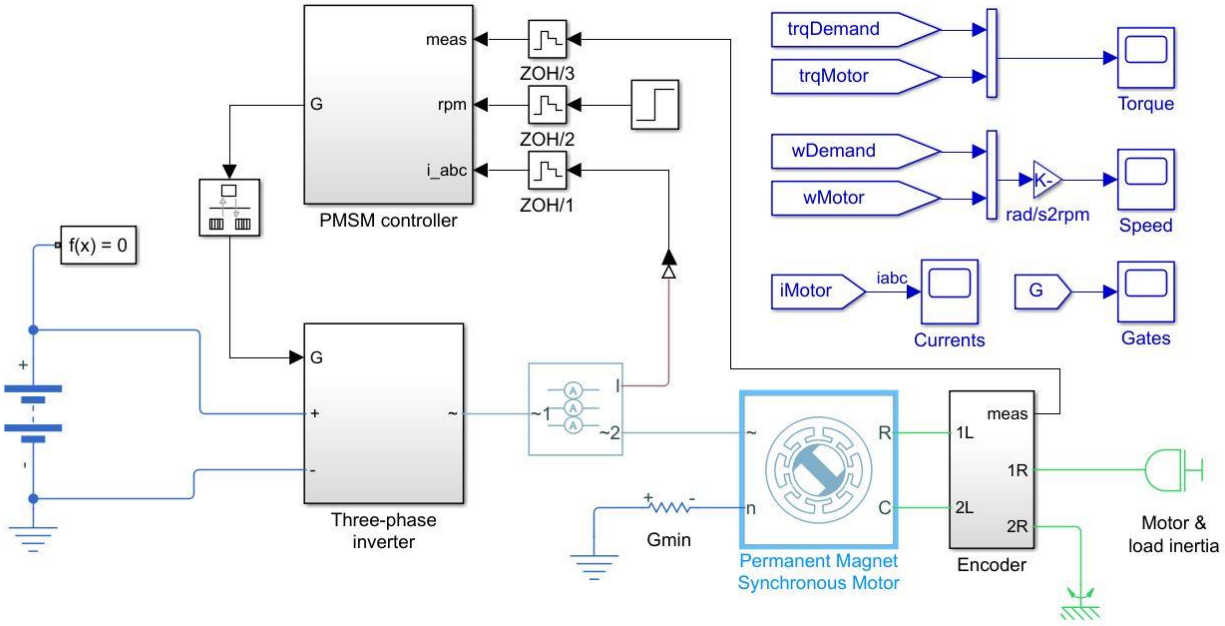

Figure 1 predefined model

The model can be used to design the PMSM controller, selecting architecture and gains to achieve the desired performances. The control strategy is based on a Field-Oriented Control that exploit a PI controller in order to follow a reference speed. Usually the reference for an electric vehicle is calculated on the torque request coming from the driver, while we considered a speed reference because we wanted to model the behaviour of a driverless car.

The project is based on two main components:

1. a MATLAB script that plots the motor torque and power as function of the speed of the rotor. Moreover, we implemented the resistive torque modelled starting from the aerodynamic force as function of the actual vehicle speed
2. a SIMULINK model that takes the curves as input in order to simulate the behaviour of the entire system.

2 SYSTEM AND CONTROL SCHEME

2.1 PMSM modelling

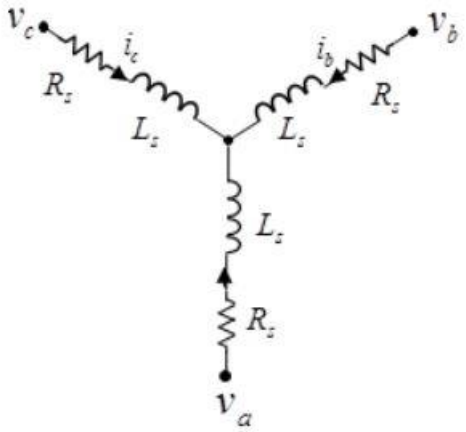

Figure 2 equivalent circuit of a PMSM in the abc reference frame

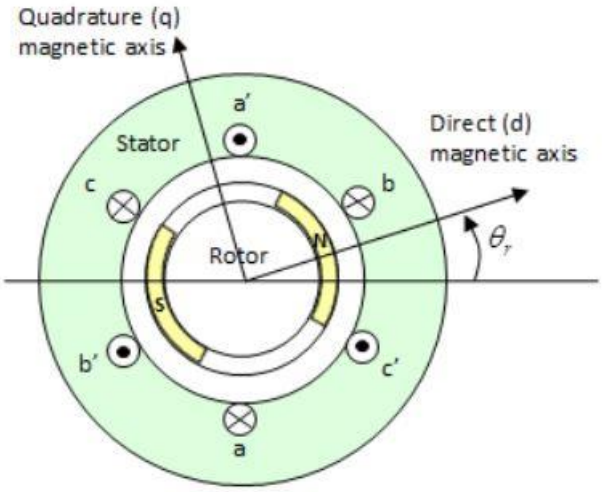

Figure 3 PMSM physical description

A PMSM is composed of three phase's stator windings and permanent magnets mounted on the rotor surface or buried inside the rotor. The electrical equations of the PM synchronous motor can be described in the rotor rotating reference frame, written in the (*dq*) rotor flux reference frame.

The mathematic model of PMSM is based on the following assumptions:

- Neglecting the saturation of armature;
- Neglecting the wastages of eddy and magnetic hysterics;
- There is no rotor damp resistance.

Accordingly, the rotor reference plane of the PMSM equivalent circuit can be:

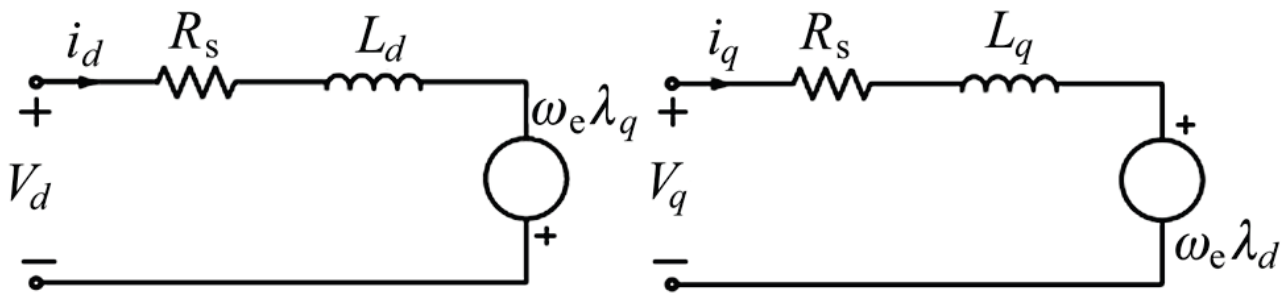

Figure 4 equivalent circuit of a PMSM in the dq reference frame

Stator *dq* equations can be written in the rotor reference plane as:

1) $V_d = R_s \cdot i_d + L_d \cdot \frac{di_d}{dt} - \omega_e \cdot \lambda_q$
2) $V_q = R_s \cdot i_q + L_q \cdot \frac{di_q}{dt} - \omega_e \cdot \lambda_d$

The dq-axis fluxes are described in the equations:

1) $\lambda_d = L_d \cdot i_d + \lambda_m$
2) $\lambda_q = L_q \cdot i_q$

The expression of λ_m represents mutual magnetic flux caused by the permanent magnet and the expression of the induced torque is:

$$T_e = \frac{3}{2} P\left(\lambda_m \cdot i_q\right) + (L_d - L_q)(i_d \cdot i_q)$$

The first term in the expression is the torque produced by the magnet and the second term is the reluctance torque due to the difference in reluctances.

The mechanical expression is provided by:

$$T_m = T_L + B \cdot \omega_r + J \cdot \frac{d\omega_r}{dx}$$

and the expression of ω_r is given by:

$$\omega_r = \int \frac{T_m - T_L - B \cdot \omega_r}{J} dt$$

In this reference frame, the maximum torque is obtained for $i_d = 0$ which corresponds to the case when the rotor and the stator fluxes are perpendicular. The drive behaviour can be adequately described by a simplified model expressed by the mechanical torque equation.

Since we exploited the motor parameter of the original system, we assumed that the system is fully reachable and controllable, without checking the determinant of the state-space matrices.

2.2 Field-Oriented control

Vector control is a method in which the stator currents are identified as two orthogonal components that can be visualized with a vector. One component defines the magnetic flux of the motor, the other one resemble the torque.

The control system of the drive calculates the corresponding current component references from the flux and torque references given by the drive's speed control. Typically, proportional-integral (PI) controllers are used to keep the measured current components at their reference values. The pulse-width modulation of the variable-frequency drive defines the transistor switching according to the stator voltage references that are the output of the PI current controllers.

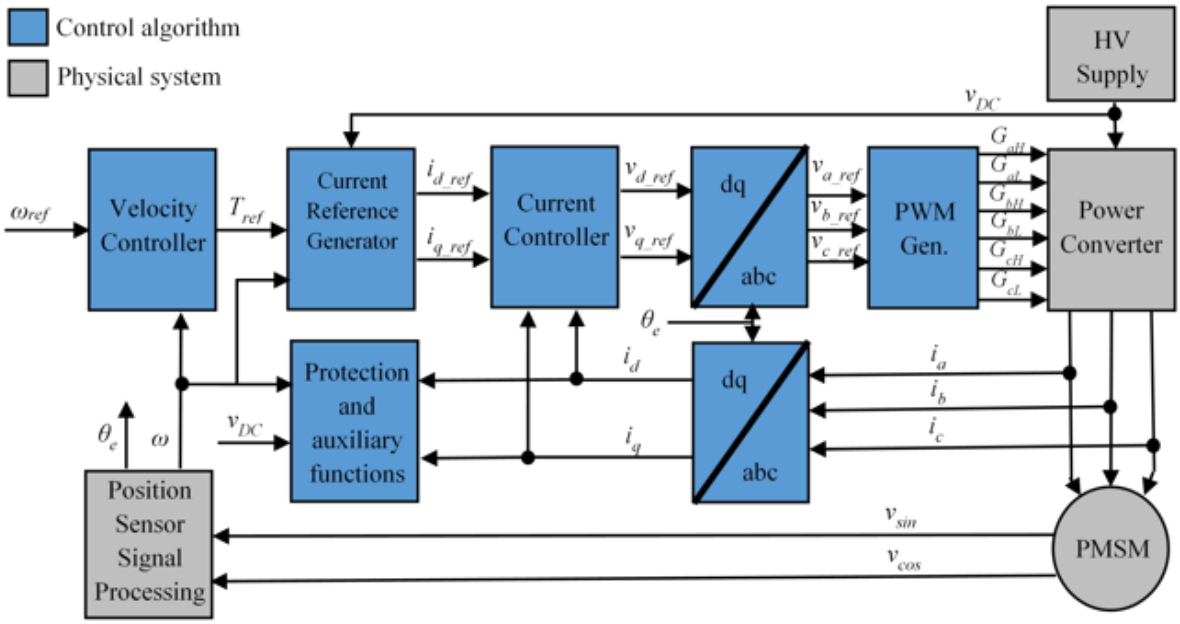

Figure 5 FOC scheme

As we chose to implement a velocity control, in the system there are two different control loops:

- Outer loop -> speed loop: a PI controller that sets the torque reference starting from the error between the actual and the reference speed, that is the input of the entire system
- Inner loop -> current loop: a PI controller that sets the voltage values in the dq reference frame starting from the current references generated considering a lookup table function on the torque

Than the voltages are used to generate the PWM that drives the gates of the inverter, generating the proper phase displacement on the three axes. The torque is driven with the currents, while the speed depends on the applied voltages.

2.3 PI controller

A proportional-integral controller (PI controller) is a feedback control loop mechanism which is widely used in control systems in industries. A PI controller calculates the difference between a measured process variable and a desired set-point which is considered as an error. By manipulating the variables, the error can be reduced by the controller.

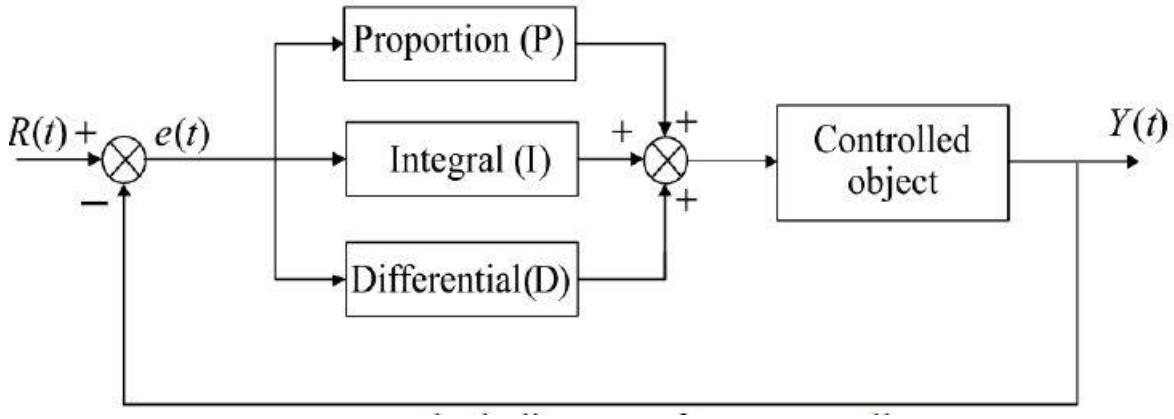

Figure 6 PI control loop

The error $e(t)$, which exists between the given value $R(t)$ and the actual output value $Y(t)$, is used by PI controller to constitute the control through the linear combination of proportion (P) and integral (I).

The conventional PI control process is written in the form of transfer function as:

$$G(s) = \frac{U(s)}{R(s)} = \frac{sK_p + K_i}{s}$$

3 SCRIPT MATLAB

First, we started to write the code in Matlab describing different parameters:

3.1 Definition of the parameters of the vehicle

```
%% DEFINITION VEHICLE VARIABLE
mass = 1725; %curb weight [kg]
cd = 0.5; %draf
frontarea = 2.4; %front area [m^2]
airdens = 1.225; %air density [kg/m^2]
gr = 7.73; %gear ratio
wheelrad = 0.317; %wheel radius [m]
```

3.2 Characterization of a PMAC Brushless electric motor

Description of the torque (N·m) and power (W) characteristic curves according to the speed of the motor (rpm). Initially, we decided to implement and study the ideal behaviour of a Brushless motor and subsequently, compare it with a not ideal one.

```
%% DEFINITION OF THE CHARACTERISTIC CURVE OF THE MOTOR

pb = 13500; %base power [W]
nb = 5000; %base speed [rpm]
tb = pb/(nb*pi/30); %base torque [Nm]
tm = 120; %maximum torque[Nm]
nm = 3500; %maximum speed at maximum torque[rpm]
nc = nb*1.01; %maximum speed at base power [rpm]
Pmax = 13500;
Tmax = 120;

motorspeed = linspace(0, 8000, 10000);

%create the torque vector
motortrq = zeros(1,length(motorspeed));
for i = 1:length(motorspeed)
    if motorspeed(i) <= nm
        motortrq(i) = tm;
    else
        if motorspeed(i) <= nb
            motortrq(i) = tb+(tm-tb)*((nb-motorspeed(i))/(nb-nm))^1.3;
        else
            if motorspeed(i) <= nc
                motortrq(i) = tb*nb/motorspeed(i);
                trq = motortrq(i);
            else
                motortrq(i) = trq*(nc/motorspeed(i))^2;
            end
        end
    end
end

motorpwr = (motortrq.*motorspeed.*pi)/30;
```

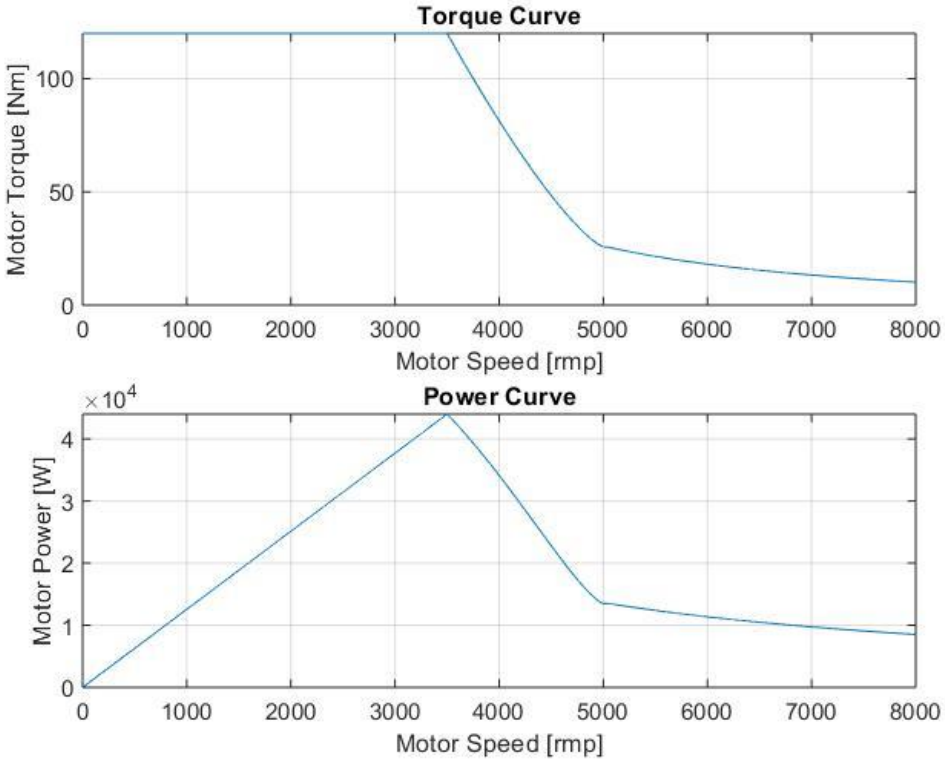

Figure 7 motor torque and power curves

3.3 Characterization of the aerodynamic force of a car

Starting from the main equations, we evaluated the aerodynamic force and power according to the speed of the vehicle such as we were able to exploit it in the Simulink's project in order to model the aerodynamic force as resistive torque applied to the crankshaft.

```
%% EVALUATION OF THE VEHICLE AERODYNAMIC FORCE
speed = linspace(0, 60, 1000); %speed range [0:0.1:60]

faero = 0.5*cd*frontarea*airdens.*speed.^2; %aerodynamic force [N]
paero = (faero.*speed)/1000; %aerodynamic power [kW]

speedkmh = 3.6.*speed;
```

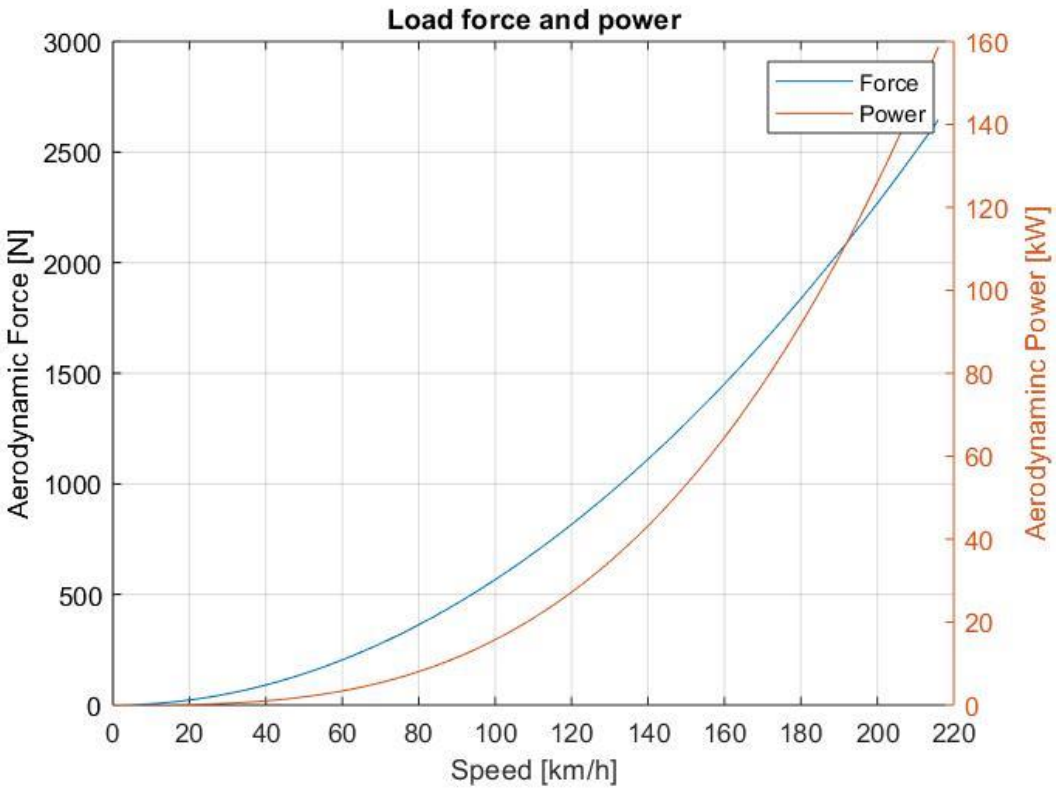

Figure 8 aerodynamic force and power curves

3.4 Creation of the speed profile

We decided to implement the reference speed as a step function of 15 m/s because it creates one of the worst situations to control. Therefore, we were able to see how our control behaves in this hard requirement. Moreover, we selected a low speed (54 Km/h) because we assumed to control a driverless hybrid vehicle, therefore we considered that the electric motor works only under a certain threshold.

```
%% CREATION OF THE REFERENCE SPEED (STEP FUNCTION)
time = linspace(1, 50, 10000);
vehiclespeed = zeros(1,length(time));
for i = 1:length(time)
    if time(i) < 1.1
        vehiclespeed(i) = 0;
    else
        vehiclespeed(i) = 15;
    end
end
```

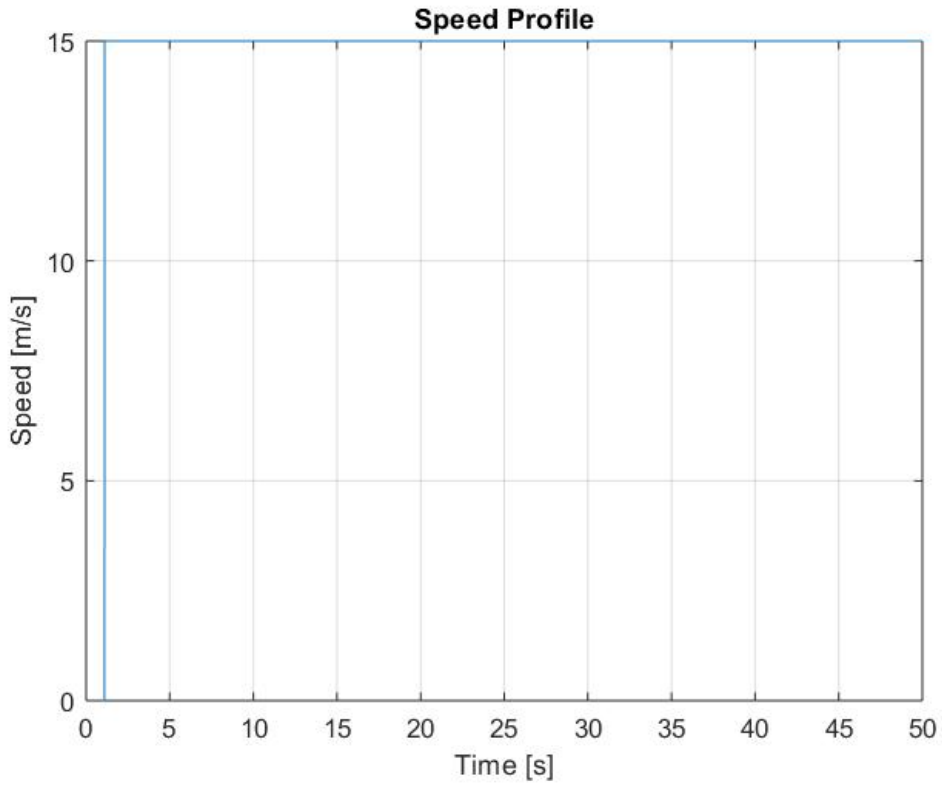

Figure 9 speed reference

4 SIMULINK MODEL

In the Simulink project, we started from an already done motor controller, the Three-Phase PMSM drive. This example is based on a Permanent Magnet Synchronous Machine, an inverter with IGBT transistor that is directly connected to the battery. Moreover, the controller implements a Field-Oriented Control (FOC) structure, which is possible to select in the parameters a velocity or torque control. We decided to choose the implementation of the velocity control because we considered a driverless car instead of a normal one.

4.1 Velocity reference of the PMSM controller

As input, we created our reference following the speed coming from the Matlab script. After that, we converted the speed of the vehicle in m/s to rad/s of the motor using the radius of the wheel, the gear ratio and the gain.

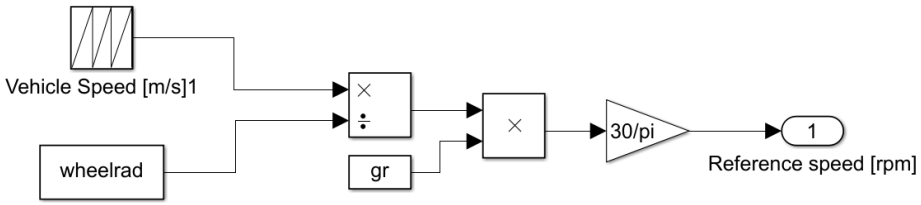

Figure 10 simulink reference speed calculation

4.2 Application of the aerodynamic force as inertia to the motor

We inserted the aerodynamic force that contrast the motor propulsion. We evaluated it, by using the actual speed of the motor inside the equation:

$$faero = 0.5 \cdot C_d \cdot frontarea * airdens * motorspeed^2$$

At the end, we convert the force into a torque and after, using the Simscape block that models an ideal torque source, we concatenated it to inertia of the cranckshaft.

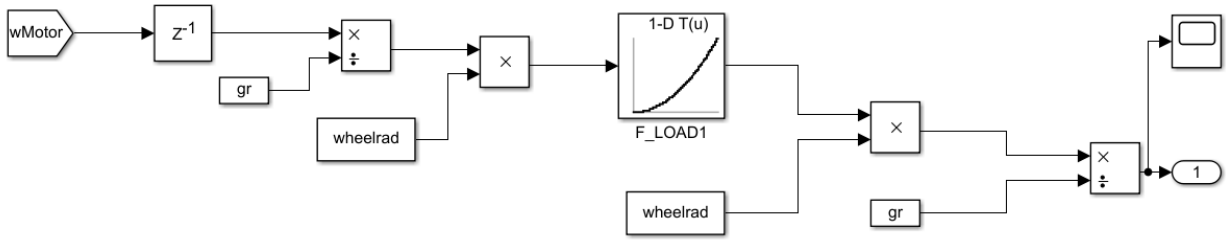

Figure 11 simulink load torque calculation

4.3 Implementation of a not ideal behaviour following and applying its characteristic torque curve

Starting from the Matlab script, we created the real feature of the torque according to the motor speed. After that, exploiting this curve from Matlab, we modified inside of the PMSM controller block (which is present already in the main project of Simulink) the Torque limiter block, by replacing Tmax (maximum torque) with the real torque coming from the characteristic curve as function of the actual motor speed.

The path to find this block is:

PMSM controller ➔ PMSM Field-Oriented Control ➔ Outer loop control ➔ PMSM Current Reference Generator ➔ Torque limiter

Here, a comparison is made in order to limit the torque request considering the lower value between the torque demand (Pmax/wMechanical) and the maximum torque (real Tmax). So, after replacing the variable Tmax as described before, the control will drive the motor following his real torque behaviour as portray in the characteristic curve.

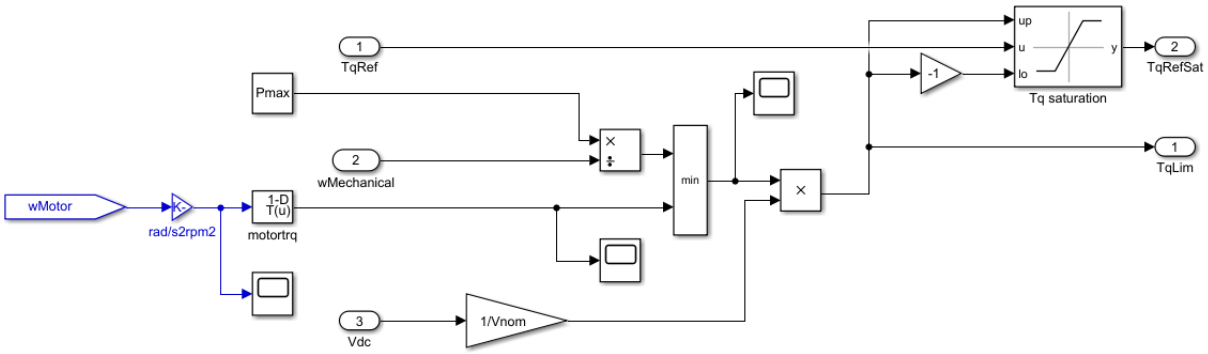

Figure 12 Simulink motor characterisation in the control loop

4.4 Modification of the control parameters

Afterwards, we modify many parameters to set properly our control and the other components following the requirements.

- Nominal DC voltage of the battery and the DC link: we put 150 V such as we were able to modify the speed with higher value.
- Higher switching frequency: we increased the fsw = 10000 in order to remove the ripple on the currents and to obtain a more precise and smooth control.
- Setting the gain values Kp and Ki of the inner loop and outer loop of the PI controller: we started by removing the rate limiter for the reference speed inside the initial control block. Afterwards, we started to tune the Kp and Ki of our system. Moreover, we put Ki = 0, such as we starting to tune just Kp to understand, in the easiest way, which was the best value of the proportional gain. At the end, we tuned and selected the Ki value to improve our control. Afterwards, we modified also the D-axis current proportional gain and Q-axis current proportional gain in order to obtain a better control.
  Finally, we set: Kp = 0.8, Ki = 18, Kpd = 1, Kid = 10, Kpq = 1, Kiq = 10.

5 CODE GENERATION

One of the main features of the Model-Based Design using Simulink is the automatic code generation for specific hardware development. In our case we chose to implement a system exploiting as target device a demo-board provided by STMiscroelectronics, i.e. STM32F407G-DISCO. In this case it has been necessary to download the proper hardware support package.

First, it's very important for the project to be deployed on a microcontroller environment that all the time constant of the simulation are the same, as it's not possible to have a variable step timing.

Another error that occur during the code generation was that MATLAB uses a variable called 'rt' as already defined value, so it has been necessary to change all the occurrences of a variable named in this way.

At this point we were able to run the Code Generation Advisor, that is a Simulink tool, which performs some checks on the model in order to understand whether it is possible to generate a coherent code starting from such a project.

The analysis is performed to have execution efficiency, RAM efficiency and to follow MISRA C guidelines, that are the specific one for an automotive application.

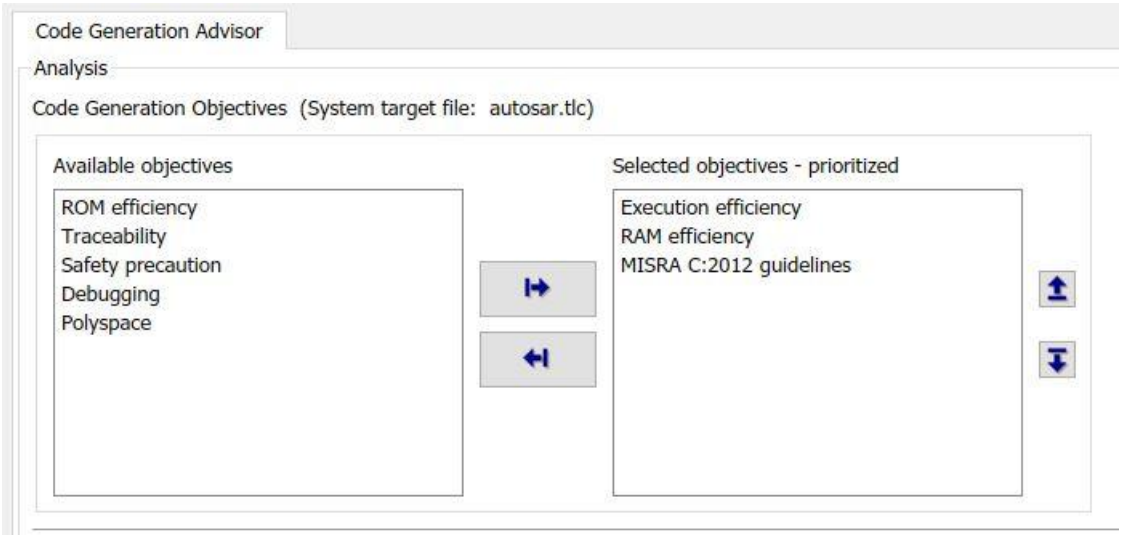

Figure 13 purposes of the analysis

In the following image it's possible to see the entire check list done during the system analysis:

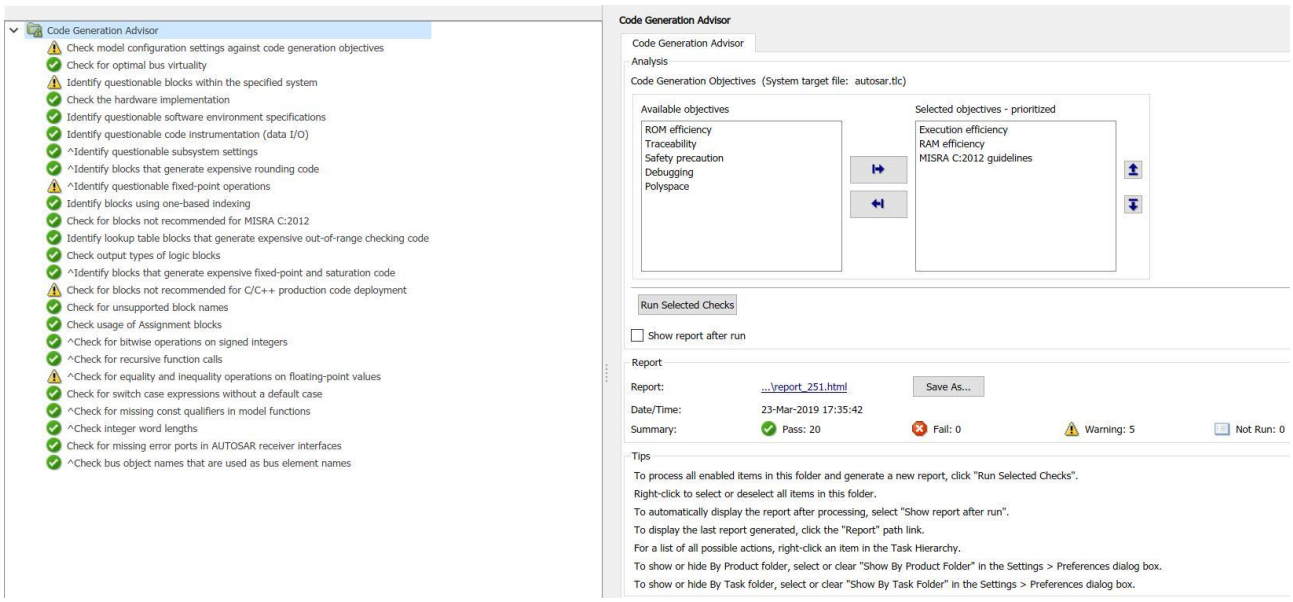

Figure 14 code generation analysis

Using the code generation tool, we were than able to generate the C code, as it's possible to verify from the report generated after that:

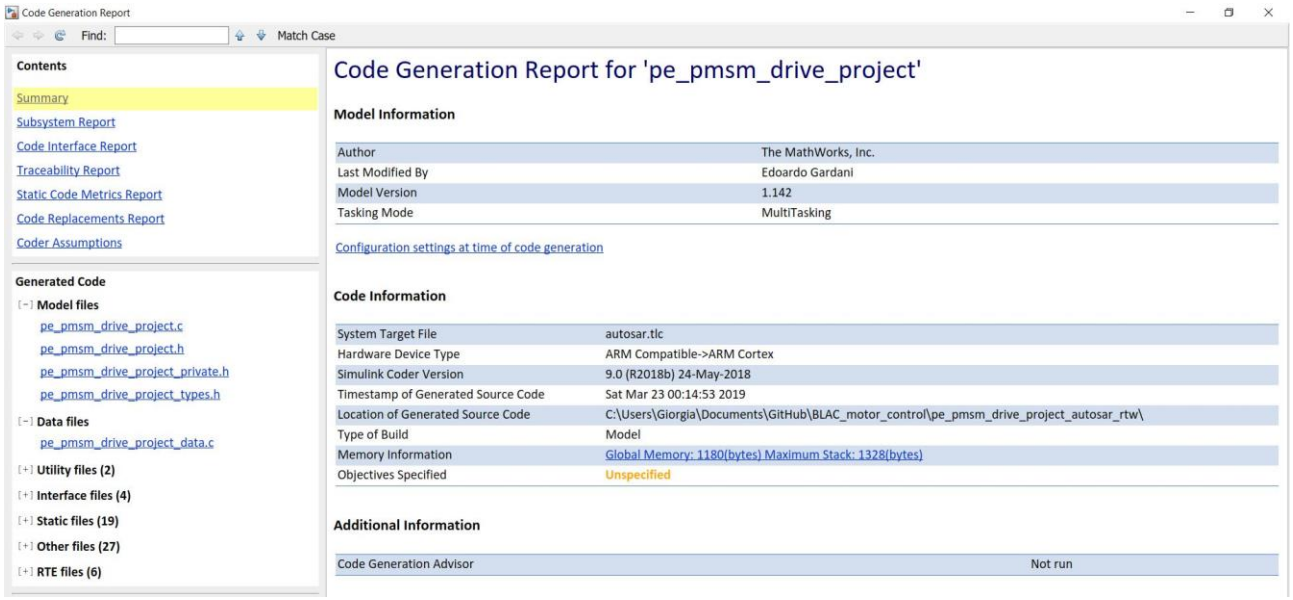

Figure 15 code generation report

6 CONCLUSION

Thanks to the study done on various graphs, we were able to set and evaluate the accurancy of our control. We started from the graph that compares the Demanded and Achieved torque, where it is possible to see a perfect overlap between them, with a shape that has at the beginning a high pick that reach the maximum torque and then it decreases as a hyperbole. We could reach this behaviour by tuning the proportional and integral gain of the Outer and Inner Loop, up to have a smooth demand torque and a perfect overlap between the two curves.

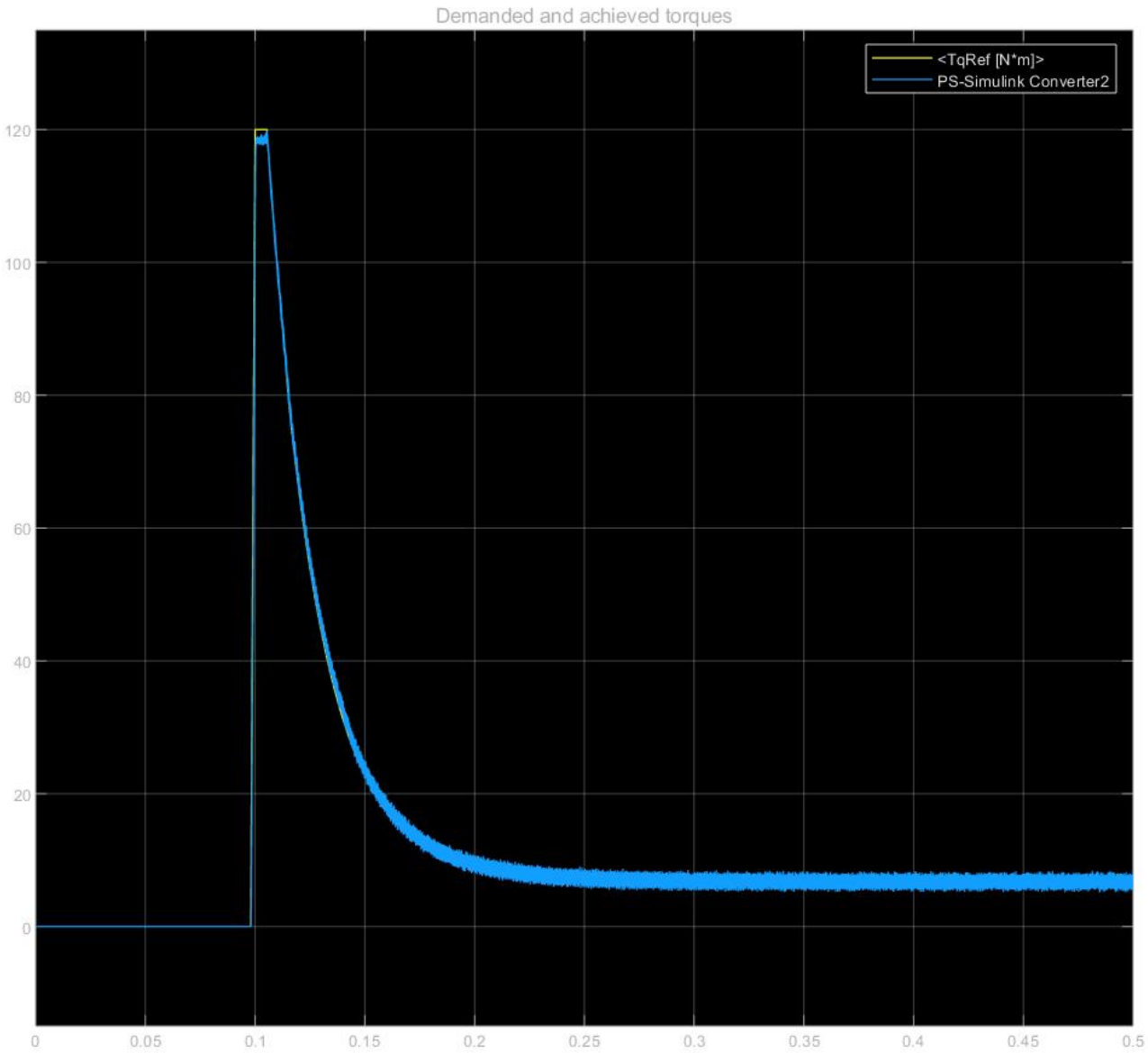

Figure 16 torque demand and request

Another useful graph which we used to understand the accurancy of our control, is the demanded and achieved speed of the motor. We made the same previous process about the tuning of the proportional and integral gain of the outer and inner loop, such as the achieved speed reaches the demanded speed and with no oscillation in the steady-state behaviour.

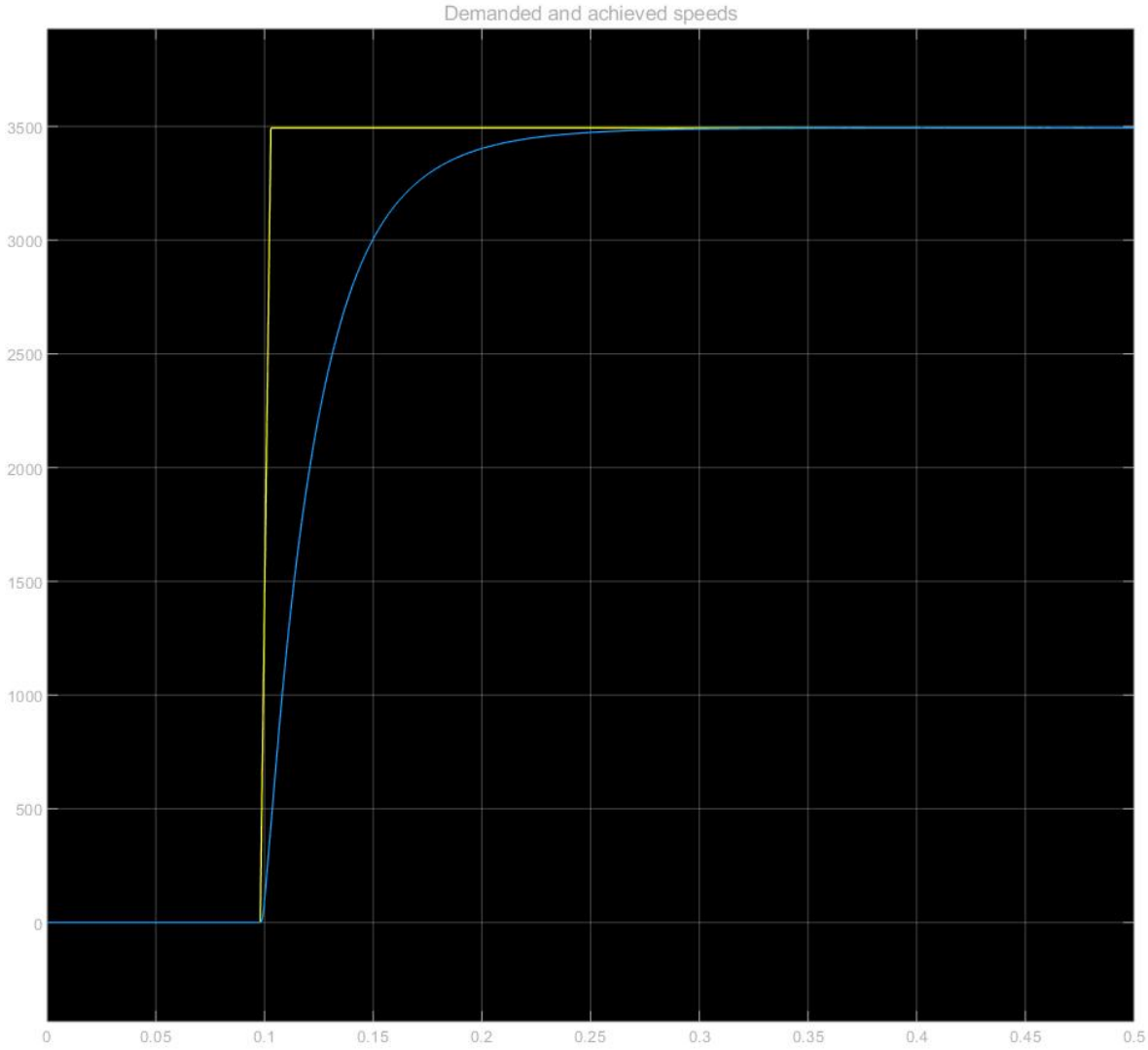

Figure 17 requested and demanded speed

The following two graphs, is a study of the motor behaviour modifying just the below parameters:

```
pb = 5000; %base power [W]
nb = 3500; %base speed [rpm]
tm = 20; %maximum torque[Nm]
nm = 2000; %maximum speed at maximum torque[rpm]
Pmax = 5000;
Tmax = 20;
```

Therefore, we obtain the following curve characteristic of the motor behaviour:

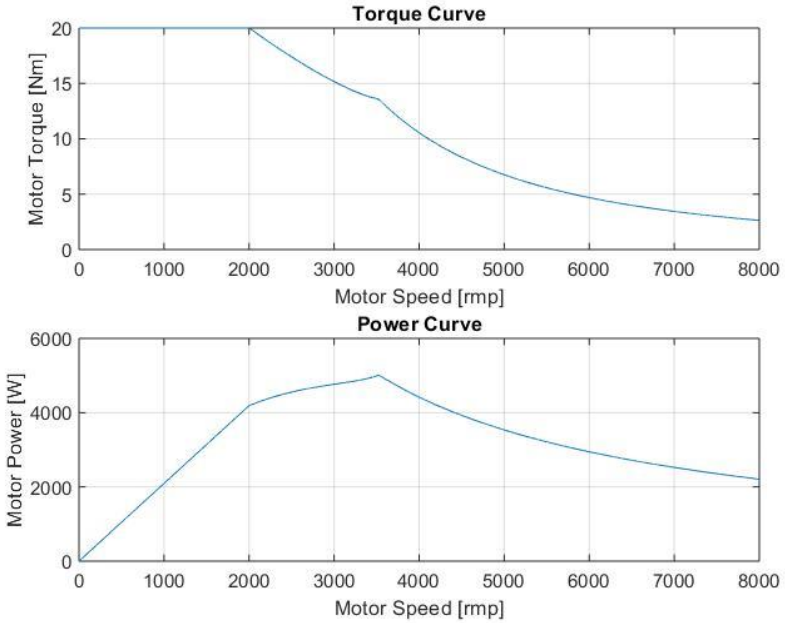

Figure 18 undersized motor characteristic

This modification has been done to see and evaluate whether the torque limiter and the control follow the curve characteristic that we defined as input in Matlab. Indeed, after having reduced a lot all parameters of the torque curve, the next graphs will show that the demanded torque (in the right figure) with respects to the requested behaviour given as input in the torque limiter.

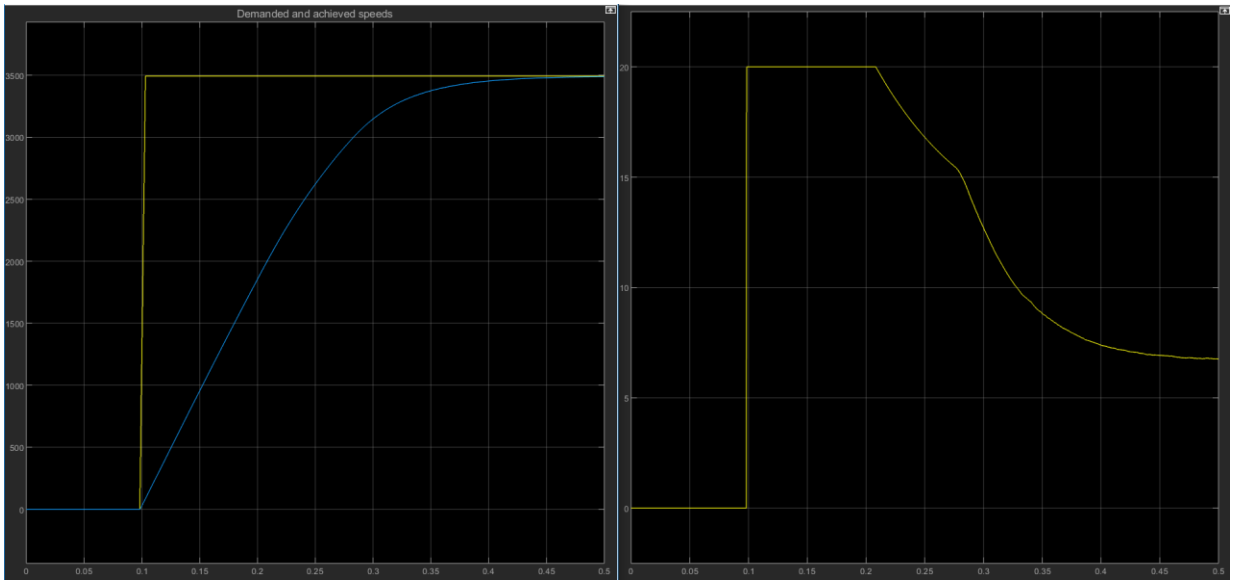

Figure 19 Speed and torque

As is possible to see in the figure below, we made the comparison between one motor with lower torque curve respect with one with much higher torque and power. As shown below, one reaches the target value of 15 m/s in less time compare to the other motor with lower torque and power.

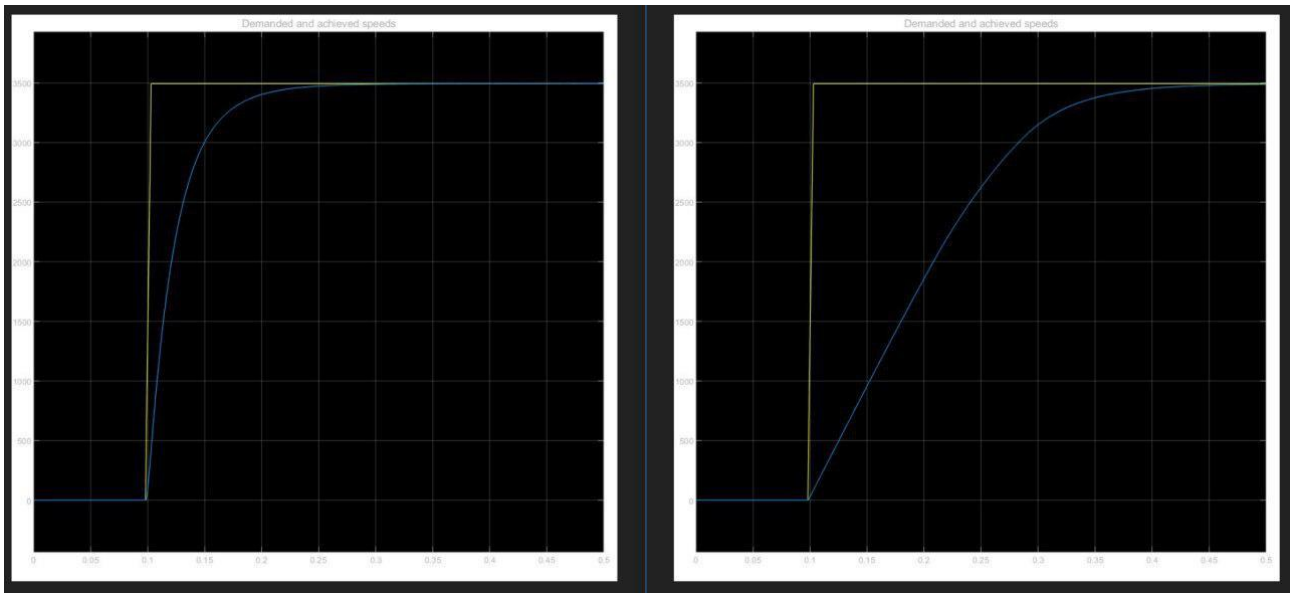

Figure 20 comparison between two different motor characteristics

6.1 Further possible improvements

- Implementation of the non-ideal behaviour of the inverter and the encoder and current sensor.
- Improvement in the non-ideal behaviour of the motor by setting the proper parameters of a real one.
- Implementation of a FEM characterization of the motor by adding the non-linearity dependence of the magnetic flux linkage with the stator currents and rotor angle.

7 REFERENCES

- The analysis of different techniques for speed control of permanent magnet synchronous motor (2015) *Mustafa Dursun, Ali Fuat Boz*
- Wikipedia: Vector control (motor)

List of figures