Motorvehicle University
of Emilia-Romagna

Advanced Automotive Electronic Engineering

Project report
of
Compliance Design of Automotive Systems

Modeling of powertrain and transmission for an electric car

Authors:
908759 Biguzzi Annachiara
889841 Di Loro Giorgio
890427 Mancini Luca
889711 Ravagli Giacomo

Professors:
Concari Carlo
Soldati Alessandro

Abstract

The goal of the project is to model and simulate the powertrain of an electric vehicle, from the motor down to the wheels. The project has been realized following the model-based design approach. The overall model has been designed in Simulink, then the controller for the electric motor was deployed on a real microcontroller, namely a Raspberry Pi 3, exploiting automatic code generation. Simulation results have been compared with the corresponding data from a real-world Volkswagen e-Golf, in order to validate the model.

Keywords: model-based design, POG modeler, PMSM, processor in-the-loop

Contents

1 Process documentation

Due to the spread of the COVID-19 pandemic, we were forced to have group meetings from remote. We decided to use Microsoft Skype for this purpose. Furthermore, we established to have at least two meetings a week, so that everyone could be sufficiently updated on other members' progresses.

Here, we report the list of the meetings, highlighting the most relevant contents discussed.

- 25/02: initial brainstorming and topic definition
- 10/03: work division
- 16/03: progress alignment meeting
- 20/03: progress alignment meeting
- 23/03: group working
- 28/03: progress alignment meeting
- 29/04: group working
- 04/04: progress alignment meeting
- 10/04: group working
- 14/04: group working
- 16/04: group working
- 18/04: group working

The work was allocated in the following way:

- Annachiara: model all the forces acting on the vehicle and design a proper-sized battery
- Luca: realize the model of the fixed-gear transmission using the power-oriented graph modeling technique
- Giorgio: select motor technology and parameters
- Giacomo: realize the motor control and processor in-the-loop

In such a situation, the presence of software for web-meetings and for versioning allowed us to make progress and actually finish the work without any face-to-face meeting.

2 Hardware and software components

All the hardware and software components used to realize the project are listed in this paragraph.

2.1 Hardware

The hardware employed is rather simple: just an ordinary PC running Windows and a Raspberry. All versions of the Raspberry are suitable; in our case we used a Raspberry Pi 3 model B+. The Raspberry has been used as core component of the processor in-the-loop simulation. For that purpose a DSP would have been preferable, since it is more suitable to serve as a motor controller; however, we had no DSP at our disposal during the realization of the project. We recommend anyone willing to replicate this project to get a DSP that is supported by Simulink Embedded Coder.

2.2 Software

Software employed for the realization of the project:

- Matlab 2019b (or above)
- Simulink: Embedded Coder package, Powertrain Blockset package are required
- POG Modeler: in order to model according to POG technique, this tool available online at http://zanasi2009.ing.unimo.it has been used. It automatically outputs the POG representation and the Simulink block scheme by compiling a .txt file written by the user. For the main rules for programming in the POG Modeler platform refer to the website.

Software employed to organize work and to enable communication between team members:

- GitHub
- LaTeX: for the realization of the documentation
- Draw-io: for the realization of graphs
- Skype: for web-meetings
- Telegram: for ordinary communications between team-members

3 Model-based design

For the realization of the project we have followed the V-model, which is one of the most used approaches in model-based design. With the instrumentation we had at our disposal

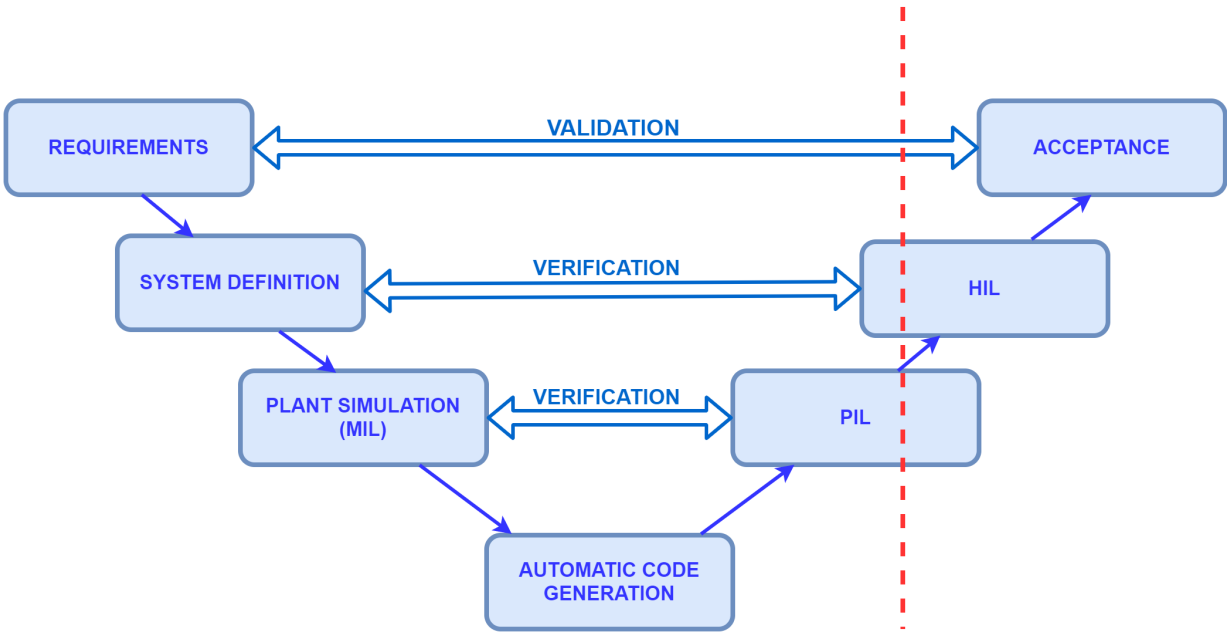

Figure 1: The V-model

we have been able to progress only up to the processor in-the-loop phase.

- **Requirements**: we evaluated model requirements, planned how to proceed with the following steps and fixed the final goals
- **System definition**: we defined a high-level design for the model
- **Plant simulation**: we realized the simulation of the plant using Simulink; the controller is also simulated in this step
- **Automatic code generation**: exploiting Simulink Embedded Coder, we were able to obtain the low-level code for the motor controller
- **PIL**: the code generated at the previous step was deployed to hardware, in our case a Raspberry Pi 3, and the processor in-the-loop simulation has been performed
- **HIL**: in this step, the plant should be simulated in a HIL module, while the motor controller should run on target hardware. We did not have the possibility to perform this step
- **Acceptance**: the real plant is realized, integration tests are performed. Again, we did not have the possibility to perform this step

4 Motor, controller and inverter

4.1 Motor

The technology of the electric motor was chosen by inspecting which types of motors have been employed in electric utility cars produced in the period between 2013 and 2019. According to [2] and to further research on car manufacturers' technical specifications, the most adopted technology was PMSM (Permanent Magnet Synchronous Motor): this is the reason why we chose this particular type of machine for our model. The next task was to find the parameters that could characterize our motor model in the most realistic way possible; this was a challenging task since complete datasheets of motors mounted on commercial vehicles are not freely available and most of the times only their performances are specified. Because of these reasons, the selection of parameters was achieved in the first place by crossing data from different sources (i.e. research papers, manufacturers' technical specifications, commercial websites) in order to understand the order of magnitude of the overall quantities that characterize an electric motor used in real vehicles and secondly by looking for the data of a specific motor that could be suitable to drive vehicles. The main sources that allowed to achieve this result are [3] and [4]. The research showed that this kind of motor should be able to generate about 75-100 kW of power and to sustain a rated torque request which is around 200-300 Nm. With this in mind the next step was to find the data for a specific motor that possessed these features in order to retrieve the numerical parameters to be used to tune our Simulink model. The quantities that define the model are, in particular, the number of pole pairs, stator resistance and inductance, the permanent flux linkage constant. The parameters of our Permanent Magnet Synchronous Motor are shown in Table 1.

Number of pole pairs	4
Stator resistance [$m\Omega$]	4.23
Stator inductance [mH]	0.391
Magnet flux linkage [Wb]	0.1039
DC linkage voltage [V]	288
Maximum speed [rpm]	4000
Maximum torque [Nm]	250
Rated power [kW]	75
Rotor inertia [kgm^2]	125
Rotor angular friction coefficient [Nm/rad]	4.924e-4

Table 1: Electric motor parameters

The automatic estimation of motor parameters would have been a better solution to the problem of finding the quantities that define the model. In fact this is how it is done if you have the real motor at your disposal, since Simulink allows to retrieve precise values for resistance, inductance and flux coefficient of the motor itself [5]. However, even without

the possibility to estimate the parameters of a real, suitable motor with this automatic tool, we were able to see that the parameters listed in Table 1 give a good representation of the real system, as will be described later.

4.2 Inverter

The three-phase inverter was introduced in order to obtain a simulation that is very close to reality. In fact, we could have omitted the inverter and the simulation would still work.

The purpose of the inverter is to feed the electric motor: it converts the DC voltage from the battery into an AC one for the PMSM motor. The voltage waveform at the output of the inverter is the PWM-counterpart of the AC waveform generated by the current controller. In this way, the PMSM motor will generate the requested output.

The inverter switching frequency was selected to be 50 kHz; this choice is a compromise between switching losses and control accuracy, and is a typical value for such applications.

Since the inverter generates ripples, a filter is needed in the connection between inverter and battery. In this phase we chose to insert an ideal low-pass filter with a cut-off frequency of 75 Hz. Moving to reality, it is advisable to adopt LC as filter technology, in order to avoid unnecessary dissipation.

4.3 Motor controller

The purpose is to generate a proper voltage waveform for the PMSM motor, in order to fulfill the control goals. In a standard automotive application, the speed controller is directly represented by the driver, while the role of the motor controller is to generate the correct amount of torque as requested by the driver pushing its pedal.

We chose to employ the controller block "Surface Mount PM Controller" from Powertrain Blockset, which implements the Field-oriented control strategy. Alternatively, we could have realized one from scratches by ourselves, but we agreed that this is not a wise strategy, since that controller is already available and extensively tested. Moreover, it offers automatic calibration of PID parameters and is ready to be deployed in real hardware, since it is designed for automatic code generation.

The following parameters were selected:

- Sample time = 100 kHz;
- Bandwidth of the current regulator = 500 kHz.

The parameters for the controller were then automatically selected by Matlab, and we verified they are appropriate to drive the motor as we desired.

The control strategy adopted is explained in more details in Section 5.

5 PMSM Field-oriented control

Field-oriented control (FOC) is a common control strategy employed to maximize the instantaneous torque generation for a given electric current supplied to the phases of the permanent magnet synchronous motor, thus optimizing the torque-generating efficiency.

The torque generation capability of an electric motor is influenced by the relative angle between the stator's and rotor's flux. Torque is maximized when the two fluxes are $\frac{\pi}{2}$ apart one from the other. Field-oriented control strategy aims at modulating the currents in the armature's phases in order to maintain this condition while the rotor is spinning.

5.1 Required information to perform the control

In order to successfully implement FOC, the following information have to be acquired:

- Instantaneous value of the electric current flowing in each phase
- Rotor angular position θ_m
- Rotor angular velocity ω_m

Since the information is obtained using sensors and is sampled by a microcontroller, we have discrete data to work on. If we don't have fresh data we should use the last information we have at our disposal to perform the control. This of course affects control performances.

5.2 Useful symbols for this section

- p number of pole-pairs in the armature windings
- θ_m rotor angular position
- $\theta_{el} = p \cdot \theta_m$ electrical angle
- φ electro-magnetic flux

For convenience, in the rest of this section we will consider $p = 1$, thus $\theta_{el} = \theta_m$ and $\omega_{el} = \omega_m$.

5.3 Park transformation

The Park transformation, better known as direct-quadrature (DQ) transformation, is an important mathematical tool employed in the realization of the FOC.

Basically, in a PMSM the armature currents are AC quantities, sinusoidal at steady state. It would be very complicated to set up a control strategy directly controlling AC currents. On the other hand, we know that controlling a DC machine is simpler, mainly because the current flowing in the armature is a DC quantity. The Park transform has

been introduced to have the possibility to control a PMSM by working with DC quantities. Actually, after the DQ transform it is possible to appreciate that the equations characterizing the PMSM are very similar to the ones seen in DC motors.

The basic idea behind the DQ transform is that we can generate the same rotating electromagnetic flux if we supply three fixed coils with sinusoidal currents, or if we supply rotating coils with DC currents. Having this in mind, it is possible to perform a transformation to transform the three AC currents in two DC currents (referred to a rotating frame):

$$\begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} \xRightarrow{\text{Park transf.}} \begin{bmatrix} i_q \\ i_d \end{bmatrix}$$

Thanks to this mathematical and physical equivalence it is possible to perform the control on the fictitious i_q, i_d currents, and then apply the inverse DQ transform to get the real phase currents. Actually, the physical control action is performed by driving a three-phase inverter, so the control signal is a PWM command to open/close the switches (see Figure 2).

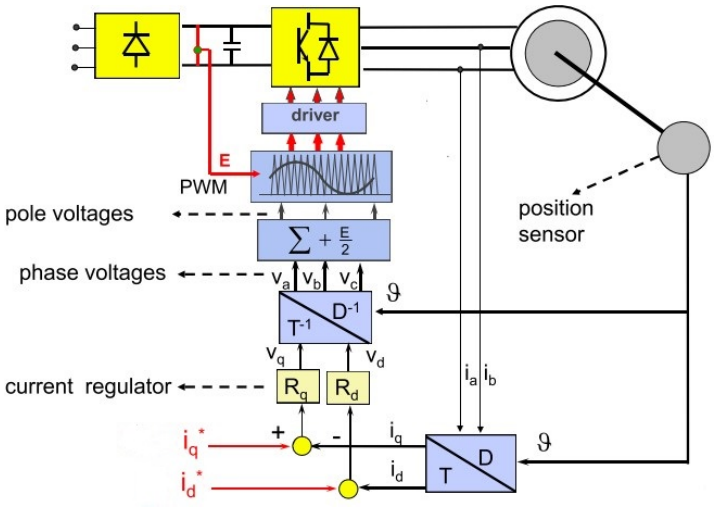

Figure 2: PMSM control scheme

5.4 Field-oriented control

As introduced before, FOC aims at maximizing the torque-generation capability of the motor. In order to do this, it is necessary to keep the armature flux in quadrature with the rotor flux (i.e. $\frac{\pi}{2}$ apart). Recall the idea the Park transform is based on. If the transformation is performed such that when $i_d = 0$ the armature flux is in quadrature with the rotor flux, we can easily understand that if we want to maximize the torque generation we need to have $i_d = 0$ always. Acting on i_d it is possible to increase the rotor speed at the

expense of a reduced torque generation capability, but it is not something we are interested in for our application.

To sum up, our control strategy aims at keeping $i_d = 0$, while tracking the control variable by modifying i_q as needed; in this way we are optimizing the power at our disposal. So, our role is to design the two controllers R_q and R_d shown in Figure 2. In our case, the two controller are PID controllers, whose parameters are automatically selected by Matlab.

6 Transmission

The mechanical section of the most general e-powertrain, whose main parts are: the rotor of the electric motor, the single-gear gearbox, the driveshaft, the differential and the semi-axles, has been designed exploiting the Power-Oriented Graph technique [6]. This technique allows to model any possible system belonging to electrical, mechanical or hydraulic domain, based on the relation between the two power variables characterizing it. The product of the power variables is the instantaneous power present in the system. One advantage of modeling according to the POG technique is that it is in fact possible to monitor the energy of the system, at each instant, in each possible section of the whole chain.

Figure 3: Standard utility-EV powertrain: the electric motor is coupled directly to the wheels through the gearbox.

6.1 Gearbox

The gearbox implemented in the project is a single-speed gearbox, with a fixed transmission ratio. The choice has been made accordingly to the most common configuration adopted in utility-BEVs nowadays. Dual-gear gearboxes are currently deployed only in high-performance electric cars, such as the Porsche Taycan, in which bigger motors with way higher power ratings are used and both requirements on acceleration and maximum speed are very strict. The transmission ratio is fixed to a value really common in this field, especially in Tesla electric cars, equal to 9.7, where the torque exiting from the motor is multiplied by 9.7 and transmitted to the drive shaft, see [7]. This value has been chosen for our application as it realizes a good trade-off between acceleration and maximum speed.

6.2 Drivetrain

The remaining transmission part, connecting the output of the gearbox to the wheels has been modeled as a rotating inertia, a rotational damper and a rotational compliance. This subsystem can be considered as the equivalent of the sequence of: driveshaft and differential, if the motor is not mounted directly on the axle of the wheels realizing the traction, semi-axles and wheels. These parameters can be neglected as they have a very low inertia and damping and are quite rigid in the case of absence of driveshaft and differential, otherwise they need to be added to the model. The presence of this additional system thus resulted in slowing down the simulation a lot, then it has not been considered for obtaining the final results. This choice has also been considered the closest to real vehicles applications, as in the majority of the electric vehicles the electric motor is mounted directly on the same axle of the driving wheels.

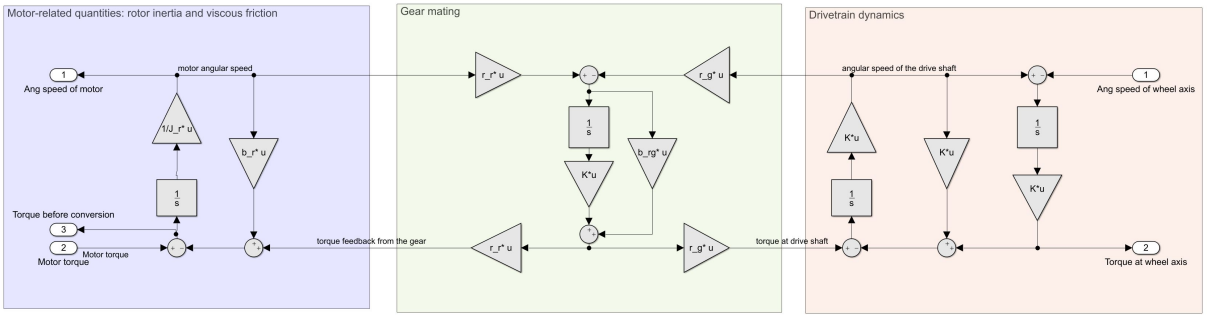

Figure 4: POG implementation of the mechanical part

7 Longitudinal Vehicle Model

We developed the longitudinal model of the vehicle using the POG (Power Oriented Graph) modeling technique, this was done in order to have a simplified model, compared to others available on Simulink, which could be understood in all its parts during all the phases of the project development.

What is considered is the impact of:

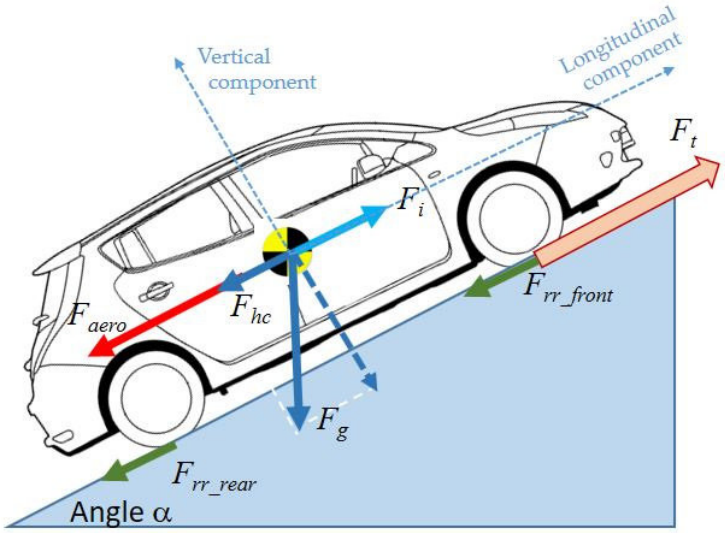

Figure 5: Longitudinal forces acting on the vehicle

- **Aerodynamic drag force**

$$F_{aero} = \frac{1}{2} C_D \cdot \rho \cdot A \cdot v_{vehicle}^2$$

Where A is the cross-section of the vehicle and $v_{vehicle}$ is its speed.
This is the main contribution between resistive forces.

- **Road friction**

$$F_{friction} = M \cdot g \cdot c_r \cdot cos(\alpha)$$

In a model in which also the lift force variation is considered the friction coefficient is not constant.

- **Road slope**

$$F_{slope} = M \cdot g \cdot sin(\alpha)$$

All the parameter values are in the Matlab file *Model_parameters.m*.

It is necessary to highlight that we did not consider a steering angle different from zero (so the vehicle goes only straight), but we added the effect of road deformation on the vehicle movement.

8 Battery Pack

The high voltage battery is one of the most important components of an electric vehicle. The first issue was related to the configuration of the battery pack. Secondly, once the specification had been defined we evaluated the proper battery cell to use. There are several chemicals used in batteries but we decided to consider only Lithium-ion cells. The main reason is that Li-ion batteries have higher specific energy [Wh/kg] and specific power [W/kg] compared with other battery types.

8.1 Battery Sizing

Starting from the assumption that we want to, theoretically, reach a distance of 300 km, we can calculate how much energy is needed to fulfill this requirement. The energy consumption is calculated from the forces opposing the vehicle's movement (road friction, drag force and slope friction) calculated using the developed longitudinal model of the vehicle. The total power P_{tot} [W] is calculated as the product between the total road forces and the vehicle speed:

$$P_{tot} = F_{tot} \cdot v_{vehicle}$$

By integrating the total power over time (for the whole duration of the cycle), we get the total energy consumption E_{tot} [J]:

$$E_{tot} = \int P_{tot} dt$$

Due to the time needed to run the cycle (total distance = 300 km), only part of the route has been simulated and in order to extract the energy [Wh] per km.

$$E_{km} = 126.3 \ Wh/km$$

This value represents the energy needed for propulsion, there would be another energy component to be considered due to the auxiliary devices (all devices supplied by the low voltage battery such as heating, ECUs ect) but for this purpose it is neglected. Moreover, we consider also an estimation of the efficiency of the powertrain $\eta_p = 0.9$ during the conversion from electrical energy to mechanical energy. Starting from the calculated value and from the efficiency we consider the resulting estimation of the average energy per km.

$$E_{avg} = E_{km} \cdot (2 - \eta_p) = 126.3 \cdot 1.1 \ Wh/km = 138.93 \ Wh/km$$

$$E_{tot} = E_{avg} \cdot target\_distance = 138.93 \cdot 300 = 41.68 \ kWh$$

To find the most suitable configuration, it is now necessary to fix the nominal voltage and total capacity whose product gives the calculated energy of the battery pack. We set the following values:

$$V_{nominal} = 550V \qquad Capacity = 75.8Ah$$

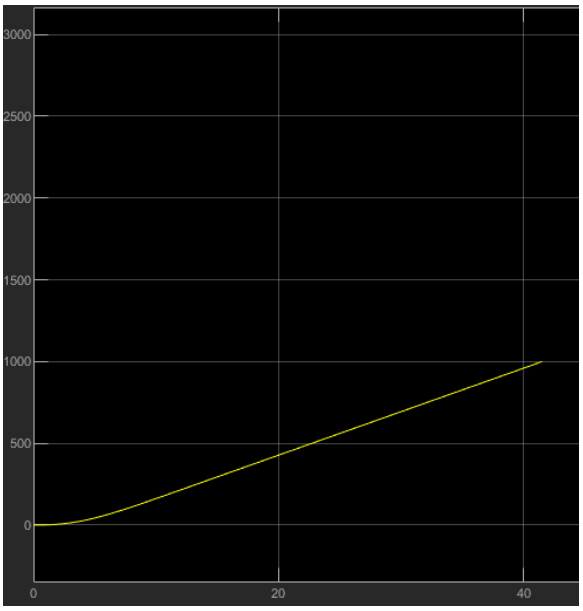

Figure 6: Distance [m]

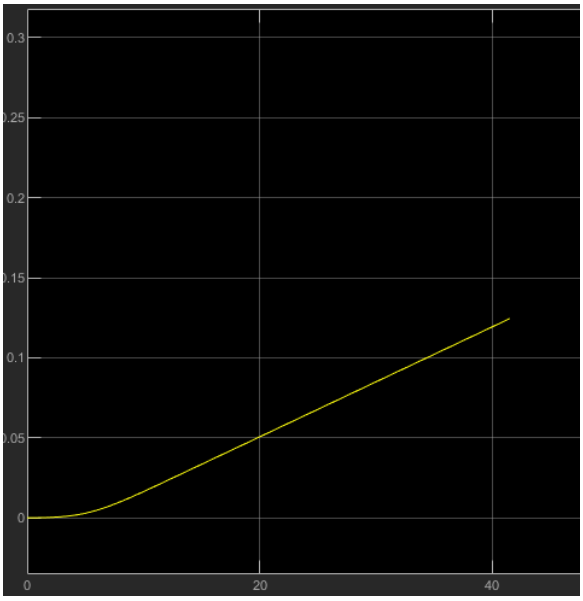

Figure 7: Energy [kWh]

The result is a battery pack with a configuration: 149s25p.
In Table 2 below all the features are summarized.

Total Energy [kWh]	41.68
Configuration	149s25p
Nomimal Voltage [V]	550
Max Voltage [V]	625.8
Min Voltage [V]	417.2
Capacity [Ah]	75.8
Max Discharge current [A]	750
Max Charge current [A]	125
Total Weight [kg]	173.585

Table 2: Battery pack parameters

8.2 Battery Cell

Once the configuration is set, we evaluate different types of cells, considering mainly the following parameters:

- **Capacity [Ah]** : the higher the capacity, the less parallel branches we have, but the maximum current at which the cell can be discharged must also be taken into account;

- **Discharge current** [**A**] : the current request from the electric motor has to be sustainable by the battery cell avoiding damage;
- **Charge current** [**A**] : in case of the development of a regenerative braking system that allows to charge the battery while braking also the charging current is an important parameter;
- **Total weight** [**Kg**]: total weight of the battery pack. During the design it's an important parameter to be considered because weight can impact on vehicle performance;
- Availability online of a datasheet that contains all the information needed.

We evaluate different types of battery cell, namely pouch, cylindrical and prismatic. Pouch and prismatic cells are widely used in automotive but it is not easy to find all the specification needed in open source datasheets. For this reason we decided to use one of the most widespread cells, Sony VTC6. It is a cylindrical shaped cell and it provides a good trade-off in term of discharge, charge current and total weight of the battery pack. The parameters of this cell are listed in Table 3.

Cell	Sony VTC6
Type	Cylindrical
Capacity [Ah]	3
Nominal Voltage[V]	3.6
Maximum Voltage [V]	4.2
Max cont. Discharge [A]	30
Charge current [A]	5
Weight [g]	46.6

Table 3: Sony VTC6 specification

The choice of the Simulink block *Datasheet battery* is due to the parameters needed for the configuration. In fact, the Simulink model based on the equivalent circuit of the cell (consisting of a resistor R_0 and one or more RC parallels) needs values for capacitance and resistance that generally are not easy to find in freely available datasheets.

9 Processor in-the-loop

The last part of the project was focused on realizing a processor in-the-loop simulation. Our goal was to deploy the motor controller block to real hardware.

Due to the spread of the COVID-19 pandemic we were forced to use the only suitable hardware we have at our disposal, namely a Raspberry Pi 3. For the purpose of controlling an electric motor a DSP would have been a better option.

First of all, we flashed a Raspbian image supported by Mathworks to the Raspberry Pi. In fact, we tried to realize a custom and optimized Linux image for the Raspberry Pi using Yocto project, we were able to compile the code locally (since the compilation is performed directly on the Raspberry) but then several errors appeared when launching the simulation. Then, in order to realize the PIL simulation we needed to properly set the "Configuration Parameters" in Simulink to generate code for the Raspberry Pi: since the connection with the microprocessor is realized using Ethernet cable, we selected the IP address and we initialized the directory where to place code on the hardware.

Once the target block is compiled, we used it to replace the corresponding model in the Simulink file, then by normally starting simulation the PIL is performed: target code runs on the Raspberry, while the plant is simulated on the PC. In this phase we noticed that the simulation speed is reduced: this may be due to the impact of communication and synchronization between the PC and the Raspberry.

It is worth to note that the PIL block has not been inserted in the Simulink file we provide. This since the file is compiled locally on the attached Raspberry Pi and then it would be an implementation-specific component. We suggest the user to realize his own block for its target, referring to [8].

10 Conclusions

The team obtained a suitable model of an electric powertrain, including the battery, an electric motor controller and inverter, a PMSM electric machine and the drivetrain, considering the resistive forces opposing to the motion of the vehicle.

Using parametric approach we were able to easily allow modifications of the main parameters in an agile way.

The simulation has fulfilled the following goals:

- Acceleration: 0-100 Km/h in 8.66 seconds
- Range: 300 Km
- Top speed: 105 Km/h

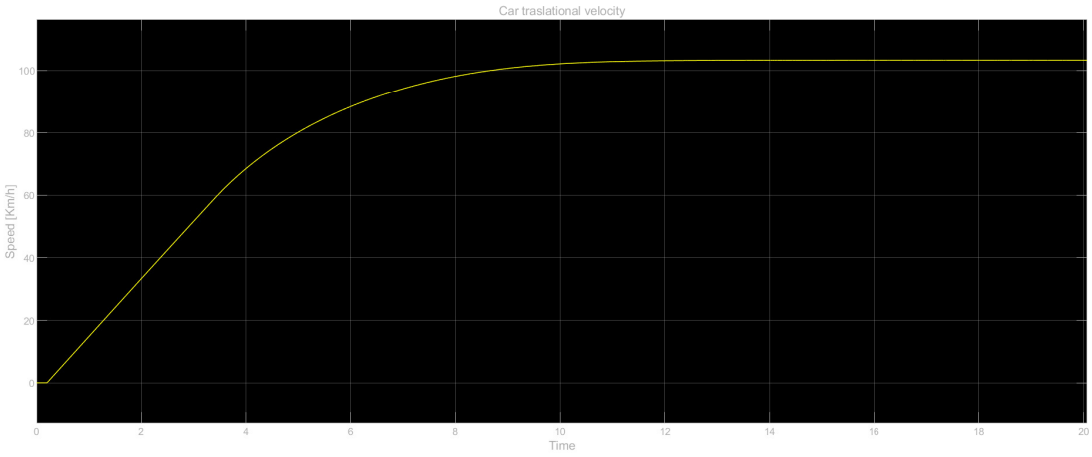

Figure 8: Vehicle speed 0-100 Km/h

Overall, the paper shows how to obtain a complete model of the system despite not having any real component at disposal. Thus, this represents a good starting point for the selection of the subsystems employed, motor and battery in particular. Fine-tuning can be performed using more accurate parameters, possibly obtained using Matlab parameter estimation capabilities to obtain the quantities from real components [5].

References

[1] Slides of Electric Drives, lessons of Prof. Domenico Casadei

[2] EV comparison: https://x-engineer.org/automotive-engineering/vehicle/electric-vehicles/ev-design-introduction/

[3] Lei Yu, Youtong Zhang, Wenqing Huang, Khaled Teffah, A Fast-Acting Diagnostic Algorithm of Insulated Gate Bipolar Transistor Open Circuit Faults for Power Inverters in Electric Vehicles

[4] Remy HVH250 motor manual

[5] Estimate motor data using Matlab: https://www.mathworks.com/matlabcentral/fileexchange/45124-motor-control-with-ti-launchpad

[6] Zanasi R., The Power-Oriented Graphs Technique: system modeling and basic properties, Vehicle Power and Propulsion Conference (VPPC), 2010 IEEE

[7] Grunditz E. A., Design and Assessment of Battery Electric Vehicle Powertrain, with Respect to Performance, Energy Consumption and Electric Motor Thermal Capability, Chalmers University of Technology, Goteborg, Sweden, 2016

[8] “How to build a PIL block”, by Biguzzi, Di Loro, Mancini, Ravagli.