Implementation in MATLAB Simulink of a basic electric vehicle

D Moldovanu[1*]

[1]Technical University of Cluj-Napoca, Automotive Engineering and Transports Department, Bd. Muncii 103-105, Cluj-Napoca, Romania

dan.moldovanu@auto.utcluj.ro

Abstract. In order to have a better understanding of influencing factors when designing an electric vehicle, the author implemented a model in MATLAB Simulink of a basic electric vehicle taking into consideration several factors, with room for improvements and addons to further studies. The purpose was to have a basic model for teaching. The model takes into consideration the basic aerodynamics of the vehicle, the electric motor (implemented via transfer function), the vehicle velocity calculation from the sum of acting forces and validation of the model. Validation of the model was done by comparing the output data like vehicle velocity with the real measurements done by the manufacturer itself. After the validation, a PI controller was implemented, and two instances were compared. The author considers that students should learn by doing, especially when simulation-based design is easy to understand.

1. Introduction

It is very important to have a model or library of models and systems/subsystems in MATLAB Simulink because they can be used for development or dimensioning or different approximations while in the design phase of the vehicle. Several papers on the issue were studied and analysed in order to have a clear perspective on the steps to take when implementing an electric vehicle.

With many electric vehicles now on the market, with an average range over 250 km on a single charge, the focus is more and more on the development and improvement of the performances for a more sustainable transport. Also because the 2010-2020 time period has been described as a tipping point period for the internal combustion engine to electric propulsion systems.

Butler et. al. [1] modelled an electric and a hybrid vehicle using MATLAB, with the V-Elph package, that offers blocks for the transmission, the internal combustion engine (for the hybrid), the battery, the different drive shafts, the induction motor and the controller, with a clever implementation of the drive cycle from a ".mat" file. The simulation concluded with the fact that all components inserted into the model need a fine tune in order to have accurate results. For all four drive cycles results were analysed for all control strategies and vehicle configuration. Lakshmi et. al. [2] implemented an electric vehicle drive simulation in order to investigate the power flow for both motoring and regeneration instances, with very detailed equations for all blocks like battery model, motor drive and different controllers. Ma [3] implemented a controller for the propulsion system of an electric vehicle to underline the waveforms of phase voltage and current, but also waveform of the power. It also presented a propulsion system

design that requires a high-power density. Sri Kaloko et. al. [4] developed a small electric vehicle and modelled it in MATLAB by taking the physical data for batteries and vehicle, in order to know the necessary battery capacity to reach a certain specification. Fan [5] used MATLAB Simulink and ADAMS to model and simulate a hybrid electric vehicle. In ADAMS, the vehicle model was developed with inputs about the chassis, suspension, driveline, tires, braking, steering and terrain info, and in MATLAB, the internal combustion engine (that was implemented using the engine maps for all throttle positions and taking into account the losses when throttle is null), battery, controller, power management and driver input were implemented. The driver controller subsystem is a complex model that considers the desired drive cycle and takes into consideration the actual vehicle speed.

2. Objectives

In this paper, the author presented a model of a basic electric vehicle implemented in MATLAB Simulink, taking into consideration the dynamics of the vehicle (Tesla Model S), a simple generated testing cycle to see the response of two different PI Controller configurations.

3. Method

First, due to the high complexity of a three-phase AC four pole induction motor, it was replaced with a brushed one for similar performances, even though AC motor rotates without contact, therefore more efficient, but an overall efficiency coefficient was taken into account to balance this.
The electric motor is a basic R-L-EMF series circuit, where R is the resistor, L is the inductor and EMF is the electromotive force generated by that motor when rotating. From this circuit, the following equation can be written:

$$V = I(t) \cdot R + L\frac{dI(t)}{dt} + E(t) \quad (1)$$

The generated torque is:

$$T(t) = K_T \cdot I(t) \quad (2)$$

K_T being the motor torque constant defined by the manufacturer.
The generated EMF can be written taking into consideration K_E (EMF coefficient of the motor) and the rotational speed of the motor $\omega(t)$:

$$E(t) = K_E \cdot \omega(t) \quad (3)$$

From the three equations, the current can be obtained with respect to the voltage and the rotational speed of the motor. After using Laplace, the motor can be implemented in MATLAB with a transfer function block with the voltage as an input and the generated torque as an output.
The next subsystem is the vehicle dynamics system, where all forces that act on the vehicle must be taken into consideration, starting with mechanical traction torque of the motor, aerodynamic drag force, roll resistance, and inertia of the vehicle.
The mechanical traction torque of the motor F_{mec} can be calculated taking into consideration the gear ratio G_r, the torque T and the diameter of the wheel, or r the radius of the wheel, but also the efficiency of the transmission eff_{tr}:

$$F_{mec} = \frac{T}{r} \cdot G_r \cdot eff_{tr} \quad (4)$$

The aerodynamic drag force can be written as:

$$D = \frac{1}{2} \cdot \rho \cdot V^2 \cdot A \cdot C_D \quad (5)$$

were ρ is the air density, V velocity of the vehicle, A frontal area of the vehicle, C_D drag coefficient.

The roll resistance is:

$$F_r = C_r \cdot m_T \cdot g \quad (6)$$

where $C_r$is the roll resistance coefficient, m_T total mass of the vehicle taking into account the mass of the passengers, and g is the gravitational acceleration.

4. Implementation in MATLAB

The constants were introduced and defined in a ".m" file so that it may be adapted to other vehicles as well. After running the m file, the data is introduced to the workspace.

```
clc
clear
close all
% definition of constants
R=5.3*10^-3;          % [ohm]
L=493*10^-9;          % [henry]
Ke=0.12;              % [Vs/rad]
Kt=0.25;              % [Nm/A]

D=0.48;               % wheel diameter [m]
r=D/2;                % wheel radius [m]
Gr=9.73;              % gear ratio [-]
%Ff=T/r*Gr            % Ff=0.45*T

rho=1.225;            % [kg/m3]
A=2.3;                % frontal area [m2]
Cd=0.24;              % drag coefficient [-]
Drag=1/2*rho*A*Cd;    % Drag

Cr=0.02;              % roll resistance coefficient (0.02 for car on dry asphalt)
mveh=2108;            % mass of the vehicle
mpass=200;            % mass of the passenger/passengers
mT=mveh+mpass;        % total mass of the vehicle
g=9.81;               % gravitational acceleration [m/s2]
Fr=Cr*mT*g;           % =413N

efftr=0.7;            % global efficiency of the transmission
Tmax=600;             % maximum torque [Nm]
Vmax=265;             % maximum velocity [km/h]
```

Figure 1. The definition of the constants for the model.

The sum of all acting forces can be written as:

$$\sum F = m_T \cdot a_v \tag{7}$$

Therefore by knowing all the forces and the mass, the acceleration can be calculated. The velocity of the vehicle appears in the 5th equation, but MATLAB allows to calculate it from the acceleration via an integration block with the chosen initial value of zero.

The calculation for the vehicle dynamics were implemented using a MATLAB function block, where the inputs were vehicle velocity and motor torque, and acceleration of the vehicle as output. The final model is presented in figure 2. The main parameters that were followed were: vehicle acceleration, vehicle velocity, motor torque, input velocity, error (calculated by subtracting the actual velocity from the input velocity), PI voltage (voltage given by the PI controller).

In order to have a realistic simulation, there are some limitations that must be inserted: motor torque limitation (by using a saturation block), because the current absorbed by the motor can be limited this way.

The maximum voltage given by the battery must also be limited. This can be done either by inserting a scope or by selecting the output limit of the controller in the output saturation tab.

The validation of the vehicle was done so that the vehicle reaches its maximum velocity (around 250 km/h) and the acceleration matches the acceleration given by the manufacturer (around 6.2 m/s^2).

As an input to the system, the vehicle velocity was chosen. The cycle that was implemented consists of a demand velocity of 50 km/h, follower by a step to 100 km/h (each for 50 seconds), follower by a 0 km/h request and then a maximum velocity demand for 200 seconds follower by motor braking for 300 seconds. When motor braking, the velocity drops gradually since the total mass of the vehicle is 2308 kg and the acting forces are the ones presented before.

For comparison, two PI controllers were compared (using a PID block but coefficients only for P and I, with D=0): PI1: P=100, I=2; and PI2: P1000, I=5; to underline the importance of the coefficients.

The Motor Model was implemented as a transfer function with numerator K_t and denominator $L \cdot s + R$, and the input takes into consideration the rotational speed of the motor $\omega(t)$ calculated from the velocity of he vehicle, knowing the radius of the wheel and the transmission ratio.

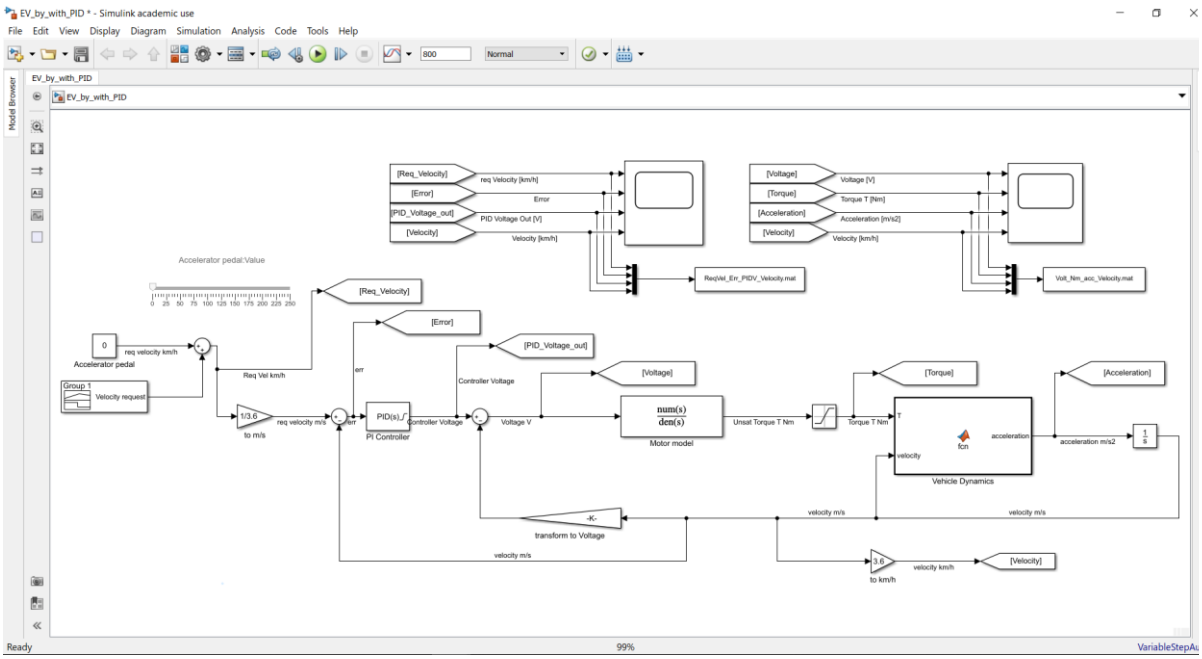

Figure 2. Simulink model implementation.

5. Results

The extracted results are presented in figures 3 and 4. The error and voltage given by the controller are presented in figure 3, and the performance results are depicted in figure 4.

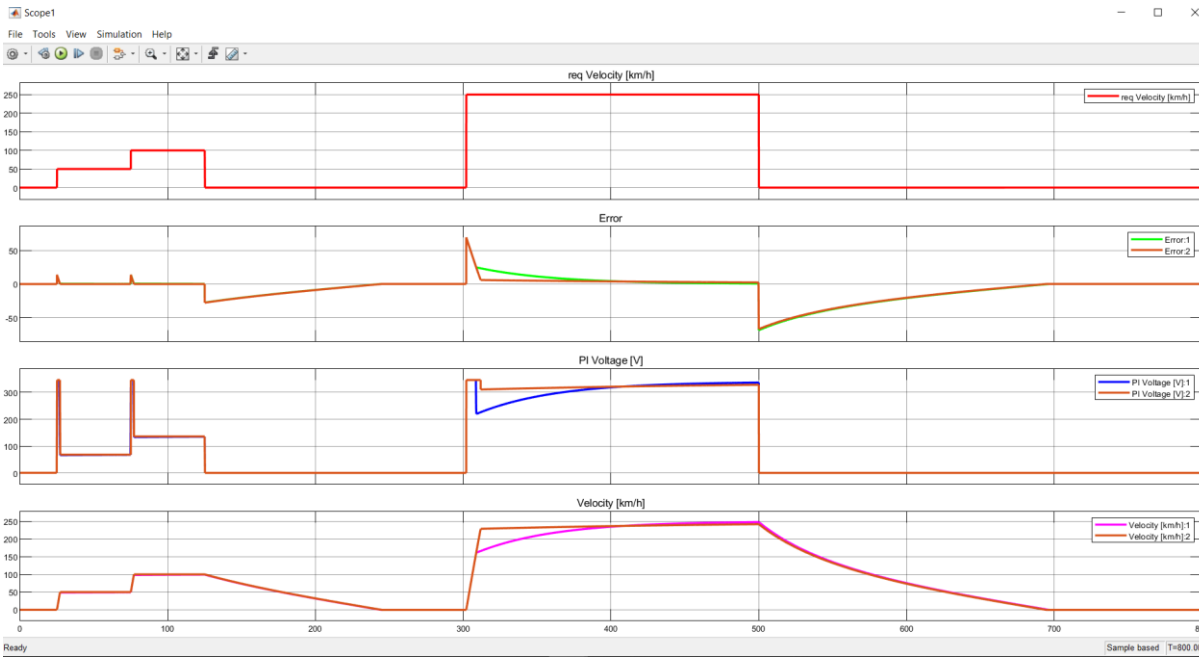

Figure 3. Error and voltage results for the two implemented controllers.

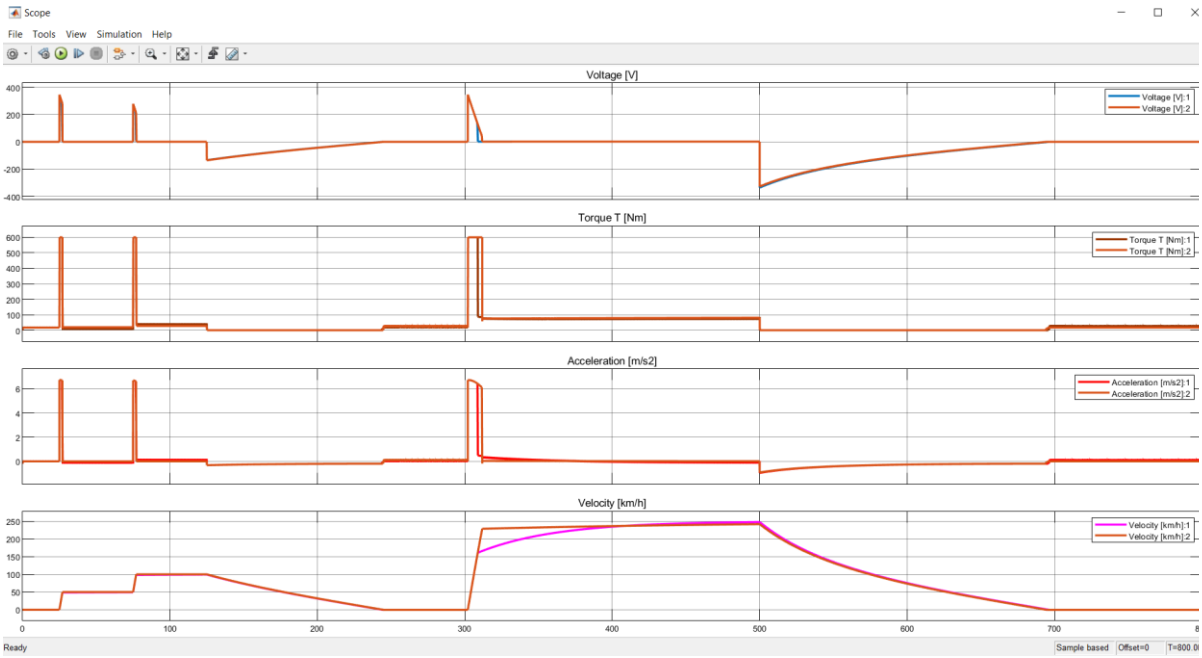

Figure 4. Performance results for the two implemented controllers

6. Conclusion

For PI1, it can be seen from figure 3 that at 300 on the time axis, the demand goes to 250 km/h, and the error starts to reduce, but because of a low proportional gain, when the error is smaller, the rise of velocity is lower, also due to the low integral coefficient.

Figure 3 underlines the fact that both controllers have a good response at low velocity demands (50 km/h errors), but at high velocity demands (step of 250 km/h), PI2 controller responds better because of the higher proportional gain. Figure 4 presents the acceleration and the torque given by the motor for both controllers. Because of the low integral and proportional coefficients, with the PI1 the motor has an early drop of torque and therefore a slower increase of velocity. Since this is a basic model, the following improvements can be considered: introduce wheel slip, modify the transfer function to a time-based model, introduce gradient of the road, filter the acceleration and implement other drive cycles.

References

[1] Butler K L, Ehsani M, Kamath P 1999 *IEEE Trans. Veh. Technol.* ***48*** *6 1770-8*

[2] Lakshmi G S, Fatima K, Madhavi B K 2017 *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON* 147-151

[3] Ma X 2002 *Proc. World Congr. Intell. Control Autom.* **1** 815-18

[4] Sri Kaloko B, Soebagio M, Hery Purnomo M 2011 *Int. J. Comput. Appl.* **24** 6 19-23

[5] Fan B S-M 2007 (thesis) Modeling and Simulation of A Hybrid Electric Vehicle Using MATLAB Simulink and ADAMS

Acknowledgements

This research was possible with the MATLAB license from the campus-wide license of the Technical University of Cluj-Napoca. Portions of this paper are based upon the research of Eliott Wertheimer. The author gratefully acknowledges the importance of his work.