Politecnico di Torino

Master of Science Thesis

Automotive Engineering

HEV modelling in Matlab-Simulink environment and control strategy design using Dynamic Programming and ECMS

Advisors
Prof. Ezio Spessa
Prof. Roberto Finesso
Dott. Ing. Federico Miretti
Dott. Ing. Alessia Musa

Candidate
Michele Giulio

March 2021

Abstract

The aim of this master thesis work is to create a working model of a Hybrid Electric Vehicle architecture on SIMULINK platform, using SIMSCAPE add-on, which allows to easily deal with multi-domain environments. The architecture selected is a P2 Parallel Hybrid vehicle, and the main idea followed during the thesis work has been to create a model with a high compatibility level with respect to the already known Optimal Layout Tool, previously developed by other colleagues of Politecnico di Torino. In order to run a "gradual complexity" philosophy, a working Battery Electric Vehicle (BEV) model, and a working Conventional Vehicle (CV) model are first designed. At the end, a third Hybrid Electric Vehicle (HEV) model is created, using as much as possible components and strategies already designed in the two previous models. The thesis work starts with a brief introduction in which simulation and modeling advantages in engineering are described, and a first part is realized in collaboration with Riccardo Russo, in which some Optimal Layout Tool simulations are run to extrapolate best Hybrid Architectures for an heavy duty vehicle facing up a World Harmonized Vehicle Cycle (WHVC). In this first part, also a linear regression analysis of results is performed. The second part of the thesis is about the already cited BEV, CV, and HEV model design process. A third last part is instead focused on comparison between these 3 models realized on SIMULINK platform with the ones realized by Riccardo Russo on GT-DRIVE platform to assess how much results obtained are similar and which are the main reasons of eventual differences.

a mia madre, che è stata il veicolo,
a mio padre, che è stato il propulsore (elettrico).

Acknowledgments

I ringraziamenti di una tesi magistrale dovrebbero essere rivolti a chi ha accompagnato l'autore durante gli anni di studi.
Vorrei usare queste poche righe per ringraziare tutti coloro i quali hanno reso i 3 anni passati al Politecnico di Torino più dolci.
Vorrei cominciare da mia madre e mio padre, che hanno reso questo traguardo possibile, prima che tangibile.
Vorrei ringraziare mio fratello Alessandro, che ha rappresentato un punto di riferimento in molte delle mie scelte accademiche e non.
Vorrei ringraziare mia nonna Giovanna, che seppur non sia più qui da molti anni, ha gettato le basi perchè tutto questo accadesse, come una candela accende la miscela.
Vorrei ringraziare Bianca per essermi stata accanto, molto spesso purtroppo a distanza, durante i momenti di gioia e tristezza. Sei molto più matura di quanto sembra.
Vorrei ringraziare Riccardo, con cui ho affrontato 10 materie su 11, e senza il quale tutto questo sarebbe stato impossibile. Ripensando a tutti i momenti di studio, il tuo contributo è stato senza dubbio fondamentale.
Vorrei ringraziare Giaca e gli altri ragazzi di Torino is The New Palermo, con cui ho condiviso momenti che non dimenticherò mai.
Vorrei ringraziare Gianluchino, che seppur non credeva di apparire in questa lista, ha rappresentato un grande amico.
Vorrei ringraziare Benedetto, per il prezioso aiuto nell'imparare ad utilizzare la scrittura in LaTEX, e non solo.
Vorrei ringraziare il professor Ezio Spessa, il dott. Ing. Federico Miretti e l'ing (a breve) Alessia Musa, che mi hanno guidato ed accompagnato nella stesura di questo lavoro di tesi. Hanno mostrato pazienza e disponibilità, e hanno sempre valorizzato le mie idee.
Vorrei inoltre ringraziare tutti gli amici di Palermo: Alessandra, Giorgio, Rita, Alessia, Gabriele e tutti gli altri, con i quali ho passato splendidi momenti durante le fasi di pausa dagli studi. Spero di potere festeggiare presto con voi.
Infine grazie alla città di Palermo, che mi ha allevato e puntualmente ri-accolto, e alla città di Torino, che mi ha formato, e che spero mi ospiterà nei prossimi anni.
Questo lavoro è stato il più complesso di tutta la mia vita, ma è stato steso con dedizione e passione.

Contents

List of Figures

Chapter 1

Introduction

Use of simulation and modelling in engineering represents without any doubt one of the most important key factors. [2] Especially for what automotive powertrain sector is concerned, in which testing vehicle on the road is a non-negligible phase, computer-based representations of vehicle systems allow not only to understand their underlying behavior, but also save money and avoid wasting time. Just think that in this thesis work more than 1000 hybrid electric architectures have been tested over a driving cycle of 20km. It is easy to imagine how much effort would have been necessary if no simulations had been available.
But advantages of simulation and modelling is not only limited in saving money and wasting time. They provide an important method of analysis which is easily verified, communicated, and understood. Looking at a computer-based model, instead of a hybrid architecture, is certainly much more immediate. Across disciplines and industries, simulation modeling provides valuable solutions by giving clear insights into complex systems.
Furthermore, due to the increased complexity of hybrid vehicle technology, development of powertrains for hybrid electric vehicles can only be performed efficiently by using the best possible simulation technology. Various simulation methods can be used depending on the given boundary conditions and the specific objectives of the investigations. As the dimensioning of drivetrain components has to be addressed in a very early phase of the development process, predictive models are of particular importance. [3] Going into further details, during this thesis work two kind of model are studied and analysed.

- Backward-facing models, in which the vehicle is assumed to be able to precisely follow the demands of the driving cycle. In this case basing on the speed trace, the resultant force at the tyre contact point is computed, where it is converted into wheel torque and propagated back to the ICE via the transmission. For these reasons, a driver model is not required.

- Forward-facing models, in which instead a driver (a PI controller) is present and it gives a torque command (via pedal) to the engine/electric motors in order to follow a pre-established driving cycle. This command propagates through the transmission and final drive ratios, before ending up as torque applied at the wheels. This is then exerted to the tyre contact point. The vehicle speed which results from the applied force is propagated back through the drivetrain, and returns to the ICE as angular velocity of the crankshaft.

It results evident that in the forward-facing models the speed is not imposed on the vehicle, so there will be instant by instant a small error between the actual and the reference speed. Role of the PI driver is minimizing this error, basing on K I factors.

scheme of both forward and backward models are shown in the next image:

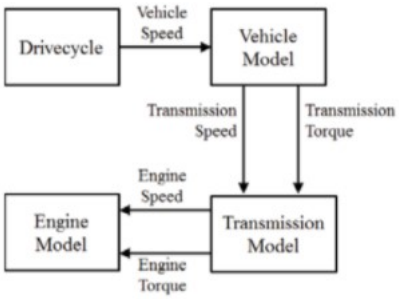

Figure 1.1: Backward-facing model basic scheme

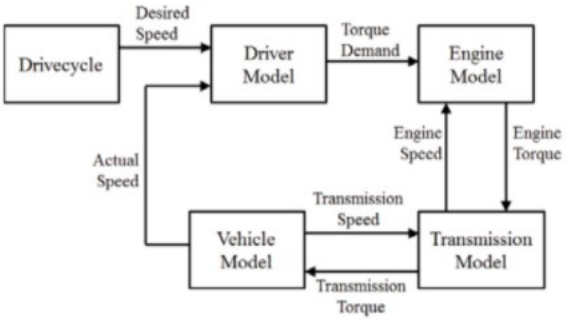

Figure 1.2: Forward-facing model basic scheme

Basically the difference between the two models is that the simulation times of backward-facing are an order of magnitude faster than the respective forward-facing. On the other hand, forward-facing models are the ones that allow to model a driver contribution to follow a cycle and so are certainly more realistic. For this reason, in the Simulink model later on designed, a forward-facing approach will be adopted.

Another important dualism that will be faced during this thesis work is the one between simulation with Optimum Layout Tool (OLT) and simulation with Simulink model. First one uses a Dynamic Programming (DP) approach, second one uses an Energy Consumption Minimization Strategy (ECMS) for the HEV controller algorithm design phase.

Chapter 2

Optimum Layout Tool

Main aim of this chapter is to illustrate the procedure used to run simulations on the Optimum Layout Tool previously developed by Politecnico di Torino, in order to select the best layout for each architecture in terms of fuel consumption and NOx over the WHVC. This simulations activity has been executed in collaboration with Riccardo Russo.
Architecture analysed during this phase have been: P0, P2, P3, P2P4, P3P4, P4. Simulations were performed first without optimization of the Aftertreatment System (ATS), then indeed they were run also with constraints related to the optimization of the Aftertreatment System.
By the way, during this chapter, only results obtained with P2 architecture and with ATS optimization will be shown, while results for other architectures will be inserted in the Appendix section. The choice of showing only this architecture is justified by the fact that P2 is the architecture that will be designed and shown on Matlab / Simulink in the next chapters.
As highlighted in the introduction chapter, OLT uses a Dynamic Programming optimizer, with the aim of finding the optimal control sequence (in terms of powerflow and gear number) to minimize emissions of CO2, and so optimizing fuel economy. Giving as input vehicle data and mission (driving cycle), it gives as a result the ranking of the layout and all parameters of the simulation.
In the case shown on this work, plausible vehicle data and a WHDC have been implemented.

2.1 P2 Architecture

As previously clarified, in this chapter, only results related to the P2 architecture with optimization of ATS are shown, even if investigation of many others architecture were executed during the thesis work and they are inserted for reference in the successive appendix section.
First of all, range of the variation list in input to the OLT is shown on the table below:

Design Parameters	Min	Max
Engine displacement (L)	3	4.4583
PE ratio	5	15
EM Peak Power (kW)	30	90
TC	4	5
FD	3	4

Figure 2.1: P2 Variation List

Fig.2.1 shows range of input parameters of the vehicle that are generated by the Variation List tool in order to create many combinations. Each of these are simulated on the OLT and a ranking Excel file is created with the results. In addition to engine displacement and electric machine power, also factors such as the Torque Coupling Device transmission ratio, the final drive and the PE ratio for the sizing of the battery (the more is the PE ratio, the less is the energy storage capacity of the battery) are taken into consideration, with the aim of investigate their influence on final CO2 emissions.
OLT outputs are shown in the next table in terms of:

- Unfeasible simulations: the ones in which for at least one time interval the power demand cannot be met by the vehicle, so the simulation is stopped and considered not valid.
- Feasible simulations: the ones in which vehicle satisfies all component speed and power constraints as well as guarantee the SOC at the end of the cycle to be equal to the initial one
- Feasible and Admissible simulations: the ones in which, in addition to the feasibility condition, other more restrictive performance conditions are respected by the vehicle.

-	#	%
Unfeasible	59	23.6%
Feasible	191	76.4%
F & adm	189	75.6%

Figure 2.2: P2 Results: Admissible, Feasible and Unfeasible simulations

Once percentages of feasible simulations are determined, further investigations focus on the generation of the scatterplot, shown on fig.2.3. This plot shows feasible layouts (in green) and unfeasible ones (in red) in function of main parameters of the variation list.

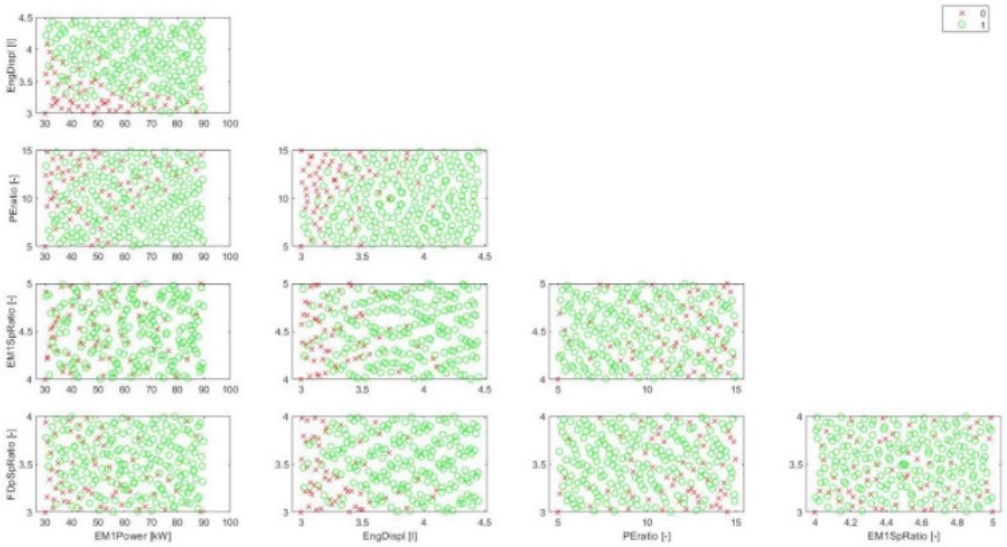

Figure 2.3: P2 Results: ScatterPlot

Scatterplot helps to immediately identify main parameters that make the simulations unfeasible. Basing on discerning some trends of green and red colours in particular zones, it is possible to get some conclusions. For example looking at the results it is evident that for range of small values of engine displacements, simulations result mainly unfeasible.
Even more interesting can be fig.2.4, that instead shows a colormap based on CO2 emissions at the end of the cycle. Again relationship between main input of the variation list are investigated. While the scatterplot helps identifying layouts unfeasible and the responsibilities of the parameters, colormap helps identifying, between feasible layouts, the ones that allow to optimize fuel consumption. Again discerning some trends in particular zones, it is possible to assess which are the main parameters that influence the fuel economy.

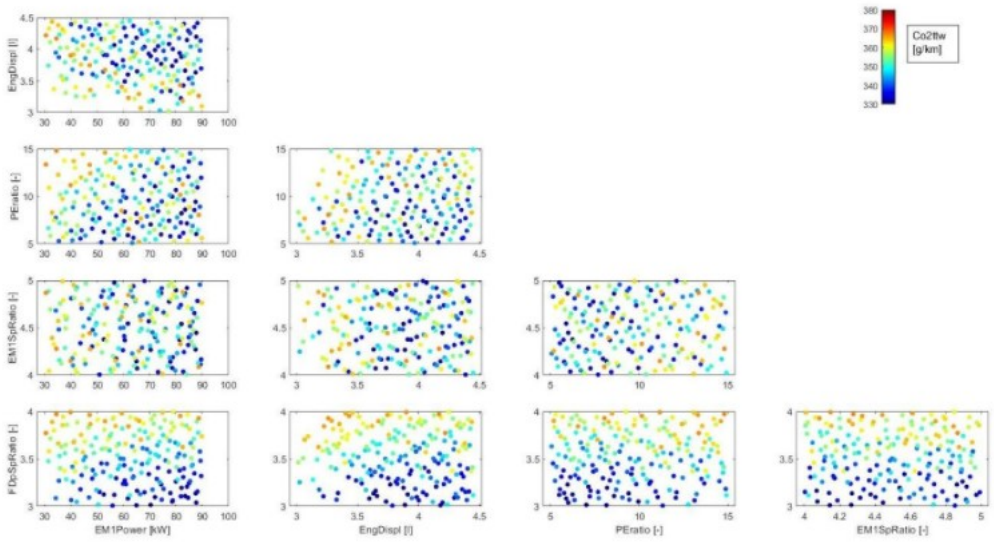

Figure 2.4: P2 Results: Colormap

For example, it seems evident that the parameters FDpSpRatio, which represents the final drive transmission ratio, has a strong influence on fuel consumption. Too much higher values cause higher emissions of CO2.

Before to investigate deeply this aspect which can lead to interesting results, it is worth to show which are the best and worst combinations of main parameters in input in terms of CO2 emissions.

EngDispl [l]	PEratio [-]	EM1Power [kW]	EM1SpRatio [-]	FDpSpRatio [-]	Co2ttw [g/km]	NOx [g/kWh]	PI [-]	TCO [k€]
3.877	8.203	82.031	4.539	3.195	339.052	0.047	46.917	346.805
3.661	8.594	83.438	4.859	3.109	339.538	0.048	43.583	345.824
3.889	7.031	70.313	4.266	3.016	339.549	0.042	43.667	346.625
3.769	5.977	74.297	4.863	3.230	339.888	0.047	41.750	346.351
3.632	9.414	87.422	4.270	3.137	340.034	0.048	44.250	346.047
4.196	6.016	87.656	4.445	3.039	340.276	0.040	49.167	348.886
3.649	7.266	65.156	4.070	3.164	340.327	0.048	38.167	345.357
3.911	6.250	52.500	4.375	3.125	340.341	0.050	38.833	346.030
4.145	7.148	84.141	4.918	3.207	340.431	0.043	48.583	348.164
3.735	9.961	78.516	4.074	3.051	340.440	0.046	42.667	345.712

Figure 2.5: P2 Results: Best Layout in terms of CO2 emissions

It can be interesting to observe that the best layout are all obtained for low values of the final drive, as already stated from the colormap. This can be justified by the fact that a lower value of the final drive helps the engine running on a higher efficiency zone of the map.

EngDispl [l]	PEratio [-]	EM1Power [kW]	EM1SpRatio [-]	FDpSpRatio [-]	Co2ttw [g/km]	NOx [g/kWh]	PI [-]	TCO [k€]
4.168	14.805	34.453	4.277	3.973	367.022	0.113	-1000	355.498
4.219	13.359	30.469	4.867	3.773	363.638	0.089	33.750	354.012
3.262	8.047	88.594	4.086	3.930	363.285	0.077	40.250	355.731
3.900	11.172	39.844	4.148	3.992	362.888	0.089	28.750	353.899
3.433	10.156	81.563	4.328	3.953	362.307	0.082	42.250	354.617
3.746	13.320	47.109	4.441	3.965	362.129	0.101	33.750	352.715
3.228	6.563	61.875	4.844	3.844	361.907	0.078	31.833	353.873
3.672	14.609	52.969	4.742	3.898	361.266	0.102	35.583	352.502
3.405	8.477	59.297	4.613	3.980	361.127	0.104	32.000	353.116
3.291	5.195	65.859	4.254	3.902	360.694	0.073	33.583	354.267

Figure 2.6: P2 Results: Best Layout in terms of CO2 emissions

At the same way, looking at the worst layouts in fig.2.6 in terms of CO2, it is evident that what they have in common is a higher value of the final drive. In addition to this, the worst layout is also feasible but not admissible, as shown by the -1000 value of Performance Index.
Since these OLT simulations were run with an optimization of the ATS aimed at reducing NOx emissions, best and worst 10 layouts in terms of pollutant emissions are also shown for reference.

EngDispl [l]	PEratio [-]	EM1Power [kW]	EM1SpRatio [-]	FDpSpRatio [-]	Co2ttw [g/km]	NOx [g/kWh]	PI [-]	TCO [k€]
4.196	6.016	87.656	4.445	3.039	340.276	0.040	49.167	348.886
3.889	7.031	70.313	4.266	3.016	339.549	0.042	43.667	346.625
4.145	7.148	84.141	4.918	3.207	340.431	0.043	48.583	348.164
4.367	5.625	63.750	4.938	3.313	342.891	0.043	47.500	348.672
4.179	8.164	64.922	4.145	3.012	340.947	0.043	44.083	346.944
4.413	10.938	88.125	4.781	3.156	342.524	0.043	53.083	348.705
4.350	8.555	86.953	4.652	3.348	342.865	0.044	52.750	349.181
4.396	6.367	40.078	4.809	3.066	344.615	0.044	40.667	347.531
4.356	5.547	43.594	4.336	3.180	343.795	0.045	41.333	347.598
4.305	13.242	66.328	4.621	3.129	345.930	0.045	48.417	348.583

Figure 2.7: P2 Results: Best Layout in terms of NOx emissions

EngDispl [l]	PEratio [-]	EM1Power [kW]	EM1SpRatio [-]	FDpSpRatio [-]	Co2ttw [g/km]	NOx [g/kWh]	PI [-]	TCO [k€]
3.353	12.422	47.344	4.273	3.867	360.282	0.145	29.917	351.621
3.245	9.883	45.234	4.160	3.684	355.352	0.133	24.083	349.560
3.461	13.086	74.766	4.012	3.988	359.236	0.127	37.667	352.642
3.217	12.734	64.219	4.180	3.586	354.208	0.125	32.750	349.806
3.348	14.180	60.703	4.715	3.660	358.406	0.119	34.917	351.395
3.370	7.461	33.516	4.324	3.801	357.322	0.118	22.000	350.085
3.308	7.109	37.969	4.492	3.648	353.416	0.117	20.417	348.736
3.256	6.055	41.953	4.902	3.598	351.812	0.117	22.667	348.496
3.513	5.703	37.031	4.789	3.945	357.387	0.114	26.083	350.804
4.168	14.805	34.453	4.277	3.973	367.022	0.113	-1000	355.498

Figure 2.8: P2 Results: Worst Layout in terms of NOx emissions

Also in this case we can note that for lower values of final drive, results are oriented towards best values of NOx emissions, while for higher values of final drive, NOx emissions slightly increases.

2.2 Linear Regression

A possible and valid interpretation of colormap plots is the one aimed to identify key variables, those which if modified causes a precise effect (in this case the effect is increasing or decreasing CO2 emissions).
To identify key variables it is necessary to observe all plots and select ones which dots show a particular trend. For example the case in which all the dots of the same colour lie more or less in the same zone, excluding instead the one in which they are homogeneously distributed.
Observing plots in the previous section, it results evident that in P2 architectures (this trend will be confirmed also for other architectures, see appendix for more details) key parameter is "FDpSpRatio".
By the way to confirm this important information it is worth to deeply analyse phenomena performing through Matlab a linear regression analysis of the results. [4]
A proper code is created in order to find for each architecture a linear relationship between CO2 and input parameters of the variation list.
As usual, in the following only results for P2 architecture is shown and commented,

while for other architecture it is possible to look at the appendix sections.
The linear model created for the regression is the following one:

$$y = \beta_1 x_1 + \beta_2 x_2 + ...\beta_n x_n + c$$

in which y is the value of CO2 emissions in [g/km], β_n represents the proportionality factors of the predictor x_n , while c represents a constant.
In particular for Matlab code value under the "Estimate" column represents β_n, while the "Intercept" row is c value:

Linear regression model:
Co2ttw_g_km_ ~ 1 + EngDispl_l_ + PEratio___ + EM1Power_kW_ + EM1SpRatio___ + FDpSpRatio___

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	303.31	4.8569	62.45	2.8583e-126
EngDispl_l_	-3.9862	0.54156	-7.3607	5.808e-12
PEratio___	0.62013	0.070957	8.7395	1.397e-15
EM1Power_kW_	-0.11543	0.012162	-9.4912	1.1709e-17
EM1SpRatio___	0.66208	0.68944	0.96031	0.33815
FDpSpRatio___	17.019	0.72072	23.614	9.9487e-58

Number of observations: 191, Error degrees of freedom: 185
Root Mean Squared Error: 2.73
R-squared: 0.816, Adjusted R-Squared: 0.811
F-statistic vs. constant model: 164, p-value = 5.26e-66

Figure 2.9: P2 Linear Regression Model

An important parameter to look at is the R^2 value, which indicates the level of correlation of the model with respect to the original one. A table of benchmark values of R^2 is shown in the following:

R^2	Grado di correlazione
0.64 - 1.00	Alto
0.25 - 0.64	Moderato
0.04 - 0.25	Debole
0.00 - 0.04	Trascurabile

Figure 2.10: Benchmark Value of correlation factor R^2

For what P2 architecture is concerned, value of R^2 is 0.816, which indicates a high degree of correlation of the model.

In addition to this, residuals coming from the model are plotted in order to further assess the validity of the model.

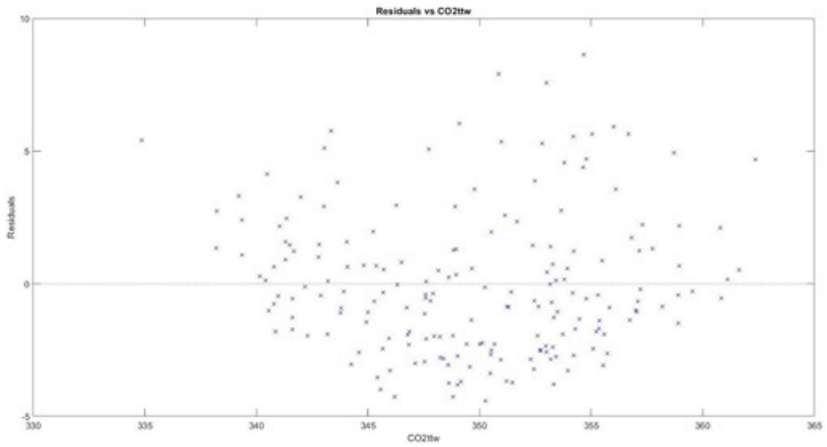

Figure 2.11: Plot of residuals

A graphical analysis of residual plot allows establishing magnitude of error introduced by the model: a random distribution, or better homogeneus, indicates an error more or less equal to zero. At the contrary a definite trend of residuals indicates a dependancy of residuals from variable CO2 and so not a completely reliable linear regression model.
In this case the distribution is by far homogeneus, so the model validated.

Aim of creating a model of linear regression is not the one of predicting accurately emissions value varying predictors, since this role is already assumed by the OLT simulations. As already said, a possible use is instead that of excluding some predictors which don't affect CO2 emissions, so that to reduce computational effort for OLT (especially for those in which ATS optimization is performed).

To do that the procedure is that of excluding once at time all predictors (except for the engine displacement, which doesn't make sense to be excluded for obvious reasons, and the electric motor power, since we are dealing with HEV), computing again for the new "reduced" linear regression model new value of R^2, of course less than the original one.
Nevertheless, in the case in which R^2 is not diminished significantly, the "reduced" model can however be considered valid, and the excluded predictor as negligible.

P2	Base	PEratio	EM1SpRatio	FDpSpRatio
R^2	0,816	0,74	0,815	0,261
Diff. %	-	9,31%	0,12%	68,01%

Figure 2.12: Linear Regression "reduced" models

From this table it is evident how for P2 architecture the "reduction" of the linear regression model excluding the EM1SpRatio wouldn't cause any significative modification to CO2 value, while as confirmed during the observation of the scatterplot and colourmap, the FDp predictor is by far the most important.
In order to assess the validity of the reduced models their residuals are plot.

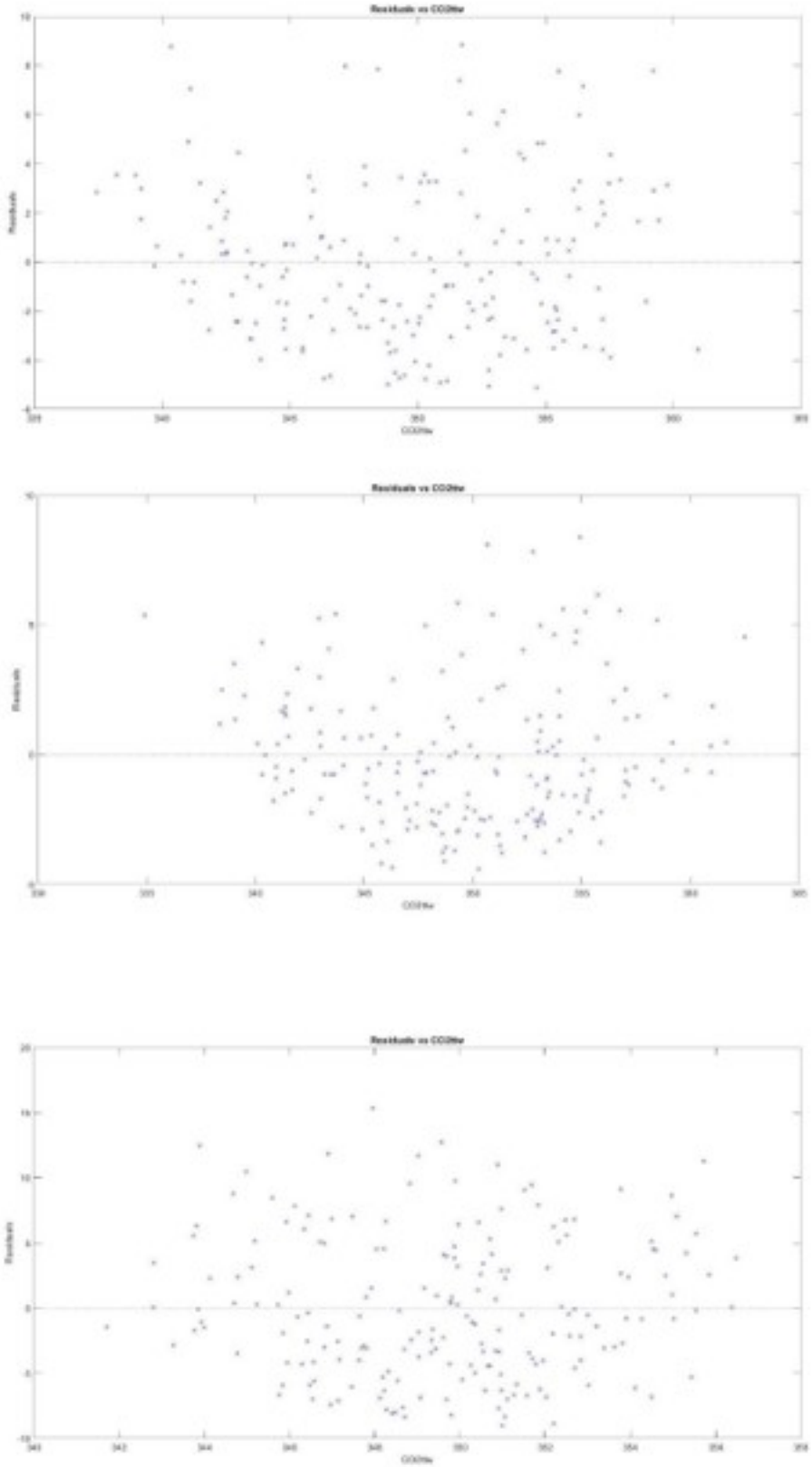

Figure 2.13: Plots of residuals removing from top to bottom: PeRatio, Em1SpRatio, FDp

All the three models are considered as valid.
In conclusion, the parameter EM1SpRatio can be considered negligible in affecting CO2 emissions for the P2 HEV in exam.

2.3 ANOVA analysis

Another alternative method, effective as well as faster, to identify negligible predictors, consists in making the analysis of variances (ANOVA) of linear models (not "reduced").
ANOVA analysis is easily executable through a proper Matlab command, and results are of immediate understanding.
For each predictor (row), column labelled as "p-value" indicates 95% significance, i.e. risk of being wrong in considering predictor as influent. Values of p-value higher than $0.05 \div 0.08$ identifies predictor as negligible.
Results of ANOVA for P2 architecture are reported in the following as verification of those obtained in previous sections.

	SumSq	DF	MeanSq	F	pValue
EngDispl_l_	404.78	1	404.78	54.18	5.808e-12
PEratio___	570.64	1	570.64	76.38	1.397e-15
EM1Power_kW_	673.01	1	673.01	90.083	1.1709e-17
EM1SpRatio___	6.8898	1	6.8898	0.9222	0.33815
FDpSpRatio___	4165.9	1	4165.9	557.6	9.9487e-58
Error	1382.1	185	7.4711		

Figure 2.14: ANOVA analysis for P2 Architecture

From fig.2.14 it is evident that also according to ANOVA analysis Em1SpRatio is a negligible parameters.

Chapter 3

Battery Electric Vehicle

Aim of this chapter is to describe what has been the procedure to create a working model of a BEV on Matlab/Simulink environment, using an add-on called "Simscape". The idea has been that of creating a model with a high compatibility level with OLT, so that to compare results of simulation coming from the latter and also exploiting the higher flexibility of Simulink. For this reasons, many choices made in the design phase have been taken so that to be as much as possible coherent with OLT model. Since this latter is a HEV model, some important differences in the architecture are by force of circumstances present. By the way this BEV model has been created in order to successively use as much as possible same components (just like the battery or the electric motor/generator subsystems) to assemble a working HEV model (described in the next chapters) which will be very similar to the one theorized by OLT.

3.1 Simscape Overview

Simscape main scope is that of extending Simulink environment for modeling of "multidomain" simulation, that are the ones in which is possible to represent more than one domain (mechanical, electric, hydraulic, pneumatic..) inside the same simulation model, and assessing their interaction. In our specific case, mechanical and electrical domains interact several times, making utilization of Simscape useful and effective. In the following fig 3.1, legend of domains present in Simscape is shown, in which colours are represented by connection of various components. In fact Simscape allows building physical component models basing on physical linking which integrate directly with block diagrams.

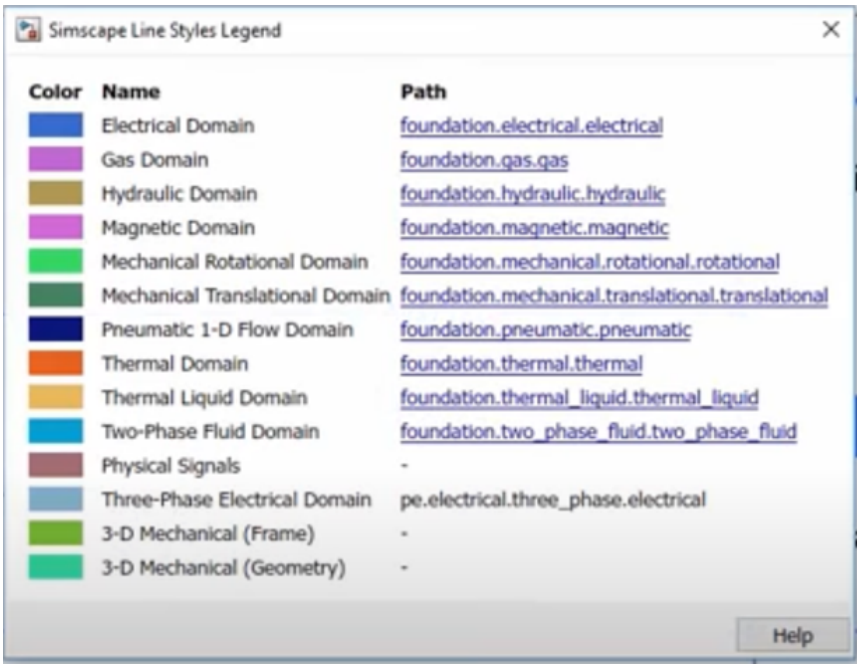

Figure 3.1: Simscape Legend of Domains [1]

Another important feature of Simscape is that it easily allows modeling of components without being necessary to derive their fundamental equations, so it strongly simplifies creation and also comprehension of models. Some "blocks" are already present in the Simscape Library that represents each one a precise component.
During this thesis work, many other add-ons have been used to build models.
In particular Simscape Driveline [5] for using some driveline blocks, StateFlow [6] for designing some state logics, Simscape Electrical [7] for accessing some electrical component in an alternative BEV model.

3.2 Model Creation

By means of the command on the matlab command window "ssc new", a new Simulink/Simscape blank model is opened, with fundamental blocks already present. In particular way they are:

1. Simulink-PS Converter, which is able to convert quantities from Simulink domain (so signals) into physical domain, to correctly interact with elements of the model.

2. PS-Simulink Converter, which is able to make the inverse operation, in order to study and plot signals, just like the speed of the vehicle or the SOC of the battery.

3. Scope, which is in charge to plot variables to being studied

4. Solver Configuration, which is a particular Simscape block that must be necessarily linked in a whichever point in the model connection, one for each present domain, in order to make possible running simulations. In the BEV model more than one of this latter is needed, since we make mechanical and electrical domain interact.

This fundamental blocks are shown in the following figure:

Figure 3.2: Simscape Fundamental Blocks

Now creation of model is going to be described, starting from the vehicle frame. Each part of the vehicle will be represented by a Simulink Subsystem, in order to keep the model ordered. Inside each of these, blocks and components and linking will be designed so to guarantee the correctness of the entire model.

Before to start describing each vehicle subsystem, it is necessary to clarify in which way data of vehicle have been correctly inserted on the Simulink/Simscape model of the BEV. An appropriate matlab file "datidelveicolo.m" is used for this mean, in which vehicle data variables are created and can also be easily modified. When the file is run data can be specified with their matlab names in the Simulink blocks.

3.2.1 Vehicle Body

Vehicle Body subsystem has a main component, the Vehicle Body block, which clearly represents the vehicle frame. This latter is used to describe a longitudinal model of the vehicle which doesn't take into account lateral forces, like the ones due to the steering-wheel rotation or other external forces, but takes into consideration longitudinal load transfers due to acceleration or deceleration of the vehicle during the driving cycle path.

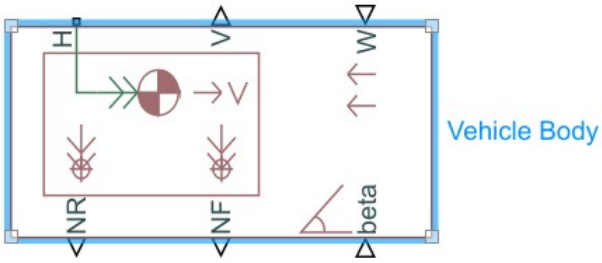

Figure 3.3: Vehicle Body Block

Numerous ports of the block indicate:

- V : output port which indicates the vehicle speed. Triangle associated means that we are dealing with a physical signal, so as already said a PS-Simulink converter will be necessary to make possible linking this port with a Scope.
- NR and NF : output ports which indicates normal loads acting on respectively rear and front wheels. In our case of a rear wheel driven vehicle, (as in OLT model), front semi-axles is not neither taken into consideration in this case.
- H: represents the physical connection responsible of the longitudinal motion of the vehicle. In easy term, the rear semi-axles. Clearly rear wheels must be connected to this port.
- W: input port which represents wind velocity in m/s. in our OLT model it is not taken into consideration.
- beta: input physical signal which models the slope of the road, which is not taken into consideration in OLT model.

Inside the block it is necessary to insert vehicle body parameter, for example number of wheels per axle or Cx aerodynamic coefficient, which are all taken from OLT Excel architecture definition file and correctly imported on matlab datidelveicolo.m.

It is now necessary to model rear wheels of the vehicle to create remaining links. Two Simple Tyre blocks are chosen. They have two ports each:

- A: it represents the physical linking coming from the transmission of the vehicle, so it is linked with the other wheel and with the transmission.
- H: it represents horizontal motion of the wheels, so it must be linked with the H port of the vehicle frame block.

About wheel parameters, wheel radius and inertia of wheels are inserted in the block.
Since the block "simple tyre" doesn't take into account Rolling Resistance of the vehicle, which in OLT model is instead modelled as constant, and since the vehicle we are dealing with is a heavy duty vehicle, so this resistance is clearly not negligible, it is a design choice to insert a proper block called "rolling resistance". This latter is able to properly model the rolling resistance in a similar way as OLT does.

For stability reason of the system, we also need to model a differential block, even if in this model it wouldn't be strictly necessary because the vehicle doesn't turn during the driving cycle maneuvers. Differential block by the way is necessary, and also allows to take into account inertia of semi-axles and of wheels of final drive, even if about the latter the transmission ratio is not here specified, but will be properly inserted later on in the transmission subsystem.

Last thing which is necessary to be modelled are the idle tyres, taken into consideration simply as two rotating objects by means of the "inertia block" in fig. 3.4

Body vehicle frame is now completed, and a subsystem is created which has as output the vehicle speed in km/hr.

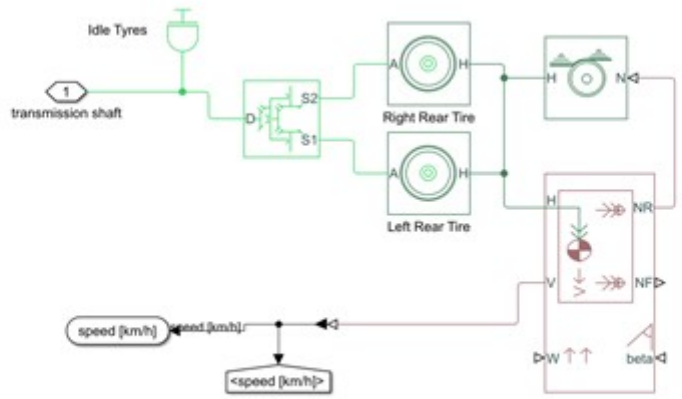

Figure 3.4: Vehicle Body Subsystem

3.2.2 Transmission

Transmission subsystem for this BEV is simply modeled as a gear pair which reduces the speed of the electric motor with a transmission ratio specifiable in datidelveicolo.m file. It represents only the final drive transmission ratio. the Simscape block "simple gear" is chosen, which has a input B and output F. B indicates the input shaft, F the output shaft. Efficiency is modeled as constant in the gear mesh, and also this value is easily specifiable.

The vehicle model is so currently composed:

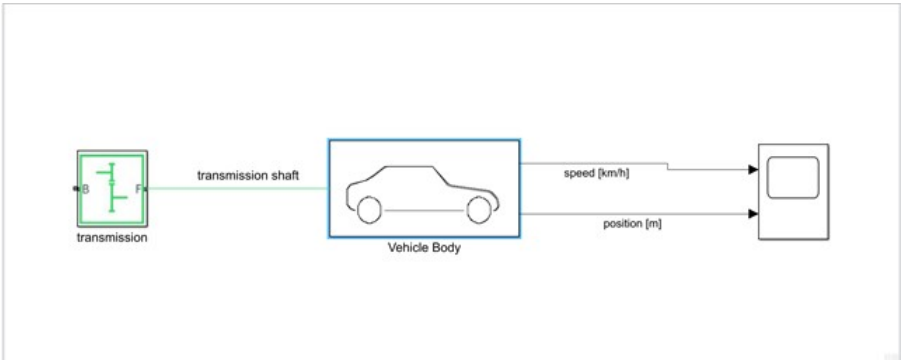

Figure 3.5: Vehicle Body + Transmission subsystems

3.2.3 Electric Motor/Generator and Controller

For what the electric motor/generator is concerned, it has been referred to a PMSM motor. This motor must be able to give traction power to the vehicle, but the model must also take into account the possibility to reverse the torque to negative values in order to recharge the battery during the braking maneuvers. More than one way can be run across in order to correctly model it, as will be shown at the end of this subsection. By the way the work is now focused to correctly design the controller of the electric motor/generator. This latter is just a subsystem present in the Electric Motor subsystem, which has on its input the rotor engine speed in rpm, coming from an RPM sensor shown in fig.3.6, and gives as output a physical signal representing

the torque command that the electric motor has to satisfy.
The RPM sensor is just modelled as an Ideal Rotational Motion Sensor block, that is linked with the output shaft of the electric motor through the port R and gives as output W a physical signal converted in rpm through the PS-Simulink converter shown. The port C is just a mechanical rotational reference that is needed and is linked to the frame of the vehicle.

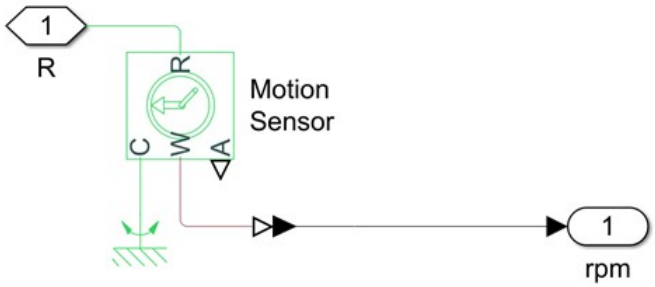

Figure 3.6: RPM Rotor Sensor

In fig.3.7 the subsystem of the controller is shown.

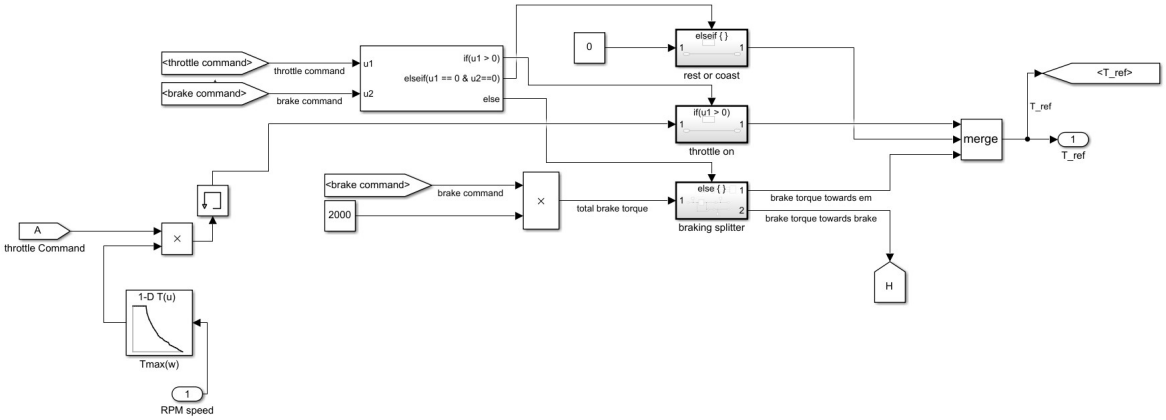

Figure 3.7: Electric Motor Controller Subsystem

Starting from the Simulink block "IF block statement [8]", it is possible to note that it is properly fed by two "From" blocks coming from a not yet shown driver subsystem. The upper one carries on a signal going from 0 to 1 that represents how much the driver is pressing the accelerator pedal. The bottom one instead carries on a signal going again from 0 to 1 but representing how much the driver is braking. Clearly during the driving cycle the two signals will be varying and in particular they will be mutually exclusive. The IF block statement discriminates three possible conditions and feeds once per time one of the three "Action Block Subsystem" [9] shown in fig.3.7. In particular:

1. If the throttle signal is positive, so it means the driver is asking the vehicle to accelerate, the Action Block Subsystem called "throttle on" will activate. this

block produces as output a Simulink signal representing the torque command which is directed towards a "merge"block and later on toward the engine. This torque value is computed scaling a percentage (depending again on the throttle command) of the maximum torque that the electric motor is able to supply at that given rpm, which comes from the rpm sensor shown previously. The maximum torque value is computed with a "Look-Up table" block, in which the electric motor maximum torque curve is correctly updated via datidelveicolo.m. This latter is supplied by OLT archives. It is possible to note from the figure that a memory block must be inserted to guarantee the stability of the system.

2. If the throttle pedal is not pressed and also the braking one is not triggered, (it means the car is at rest or in a coast phase), the "rest or coast" action subsystem will activate, returning a 0 torque command.

3. If the braking pedal is pressed, so the driver is asking the car to decelerate, the "braking splitter" action subsystem activates. This latter includes inside a braking logic accurately designed for this BEV model that will be described in the following.

Brake Logic

Vehicle is able to brake by means of two ways:

- By means of the brake system
- By means of the electric motor

In the first case braking torque will be addressed towards rear wheels inside the vehicle body subsystem and the energy will be wasted. In the second case instead the electric motor will resist to the motion of the vehicle and receiving a negative torque it will recharge the battery.
In a BEV usually both ways are run during the braking maneuvers. Complexity of braking control strategy is in imposing which percentage of braking torque addressing towards brake system, which one addressing toward the electric motor. In this BEV model the choice has been to develop a strategy according to which increasing braking torque by the driver by means of the pedal, up to a certain threshold, all this torque will be addressed toward the electric motor. If the driver asked braking torque bigger than the threshold torque, torque would be split in this way:

- electric motor would receive always the threshold braking torque
- remaining part of the braking torque would entirely be absorbed by braking system.

To implement this kind of strategy it is necessary to define on datidelveicolo.m an interpolation curve of braking torque addressed to the electric motor. This curve follows the one of the total braking torque up to a threshold value, overcome which has always the same threshold value. This trend is shown in the following figure:

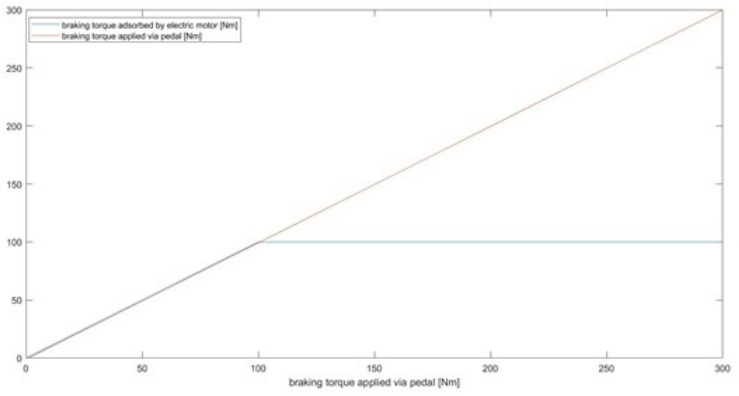

Figure 3.8: BEV Braking Torque Strategy

The threshold braking torque value is set to 100 N*m, which represents one third of the maximum torque deliverable/absorbable by the electric motor (it has a mirror shaped maximum torque curve). The reason of this choice is to not overstimulate the electric motor during the braking phases, and also verify that also the brake system is called into question. By the way this strategy will be modified in the HEV model creation chapter.

Coming back to the model, how evident in fig. 3.7, if the braking pedal is pressed, the total braking torque is computed scaling (in this case depending on the percentage of brake) a maximum constant torque value, set to 2000 N*m. This signal is the input of the braking splitter action subsystem, properly activated by the IF statement block. This block is the out-and-out controller which allows to split torque between the two braking systems. As output of this block there is the torque addressed to brake system, through a GoTo block toward braking system (not yet described) and torque addressed to electric motor, that is brought in the "merge" block and and after that in an outport that represents the output of the Controller. Fig. 3.9 shows what is inside the braking splitter:

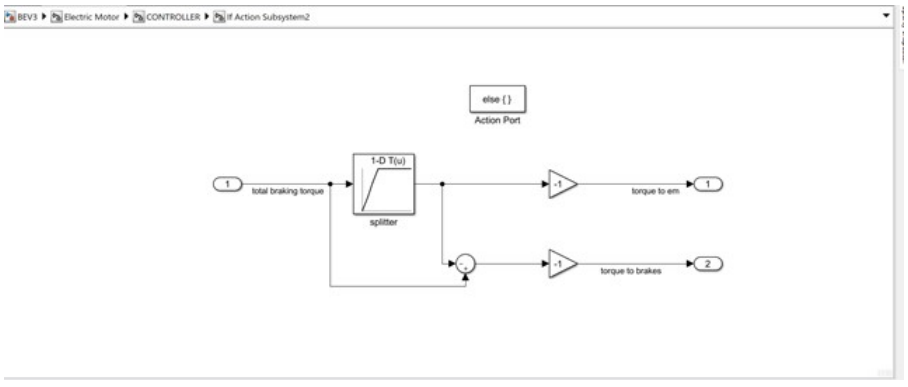

Figure 3.9: Braking Splitter Subsystem

When the total braking torque signal enters through the inport, it is "filtered" by splitter block, which is a 1-D look-up table inside which vectors representing the braking logic are implemented through matlab. In this way:

- If torque value is lower than 100 N*m, from splitter block all the torque passes, and goes straight to the electric motor.
- If instead the torque value is higher than the threshold value, from the splitter only 100 N*m goes out. The rest of the torque is computed subtracting to the total one, that coming out from splitter, and is addressed toward an output. As previously highlighted, this torque will be successively sent to the braking system.

At the end the controller in the Electric Motor subsystem is so composed:

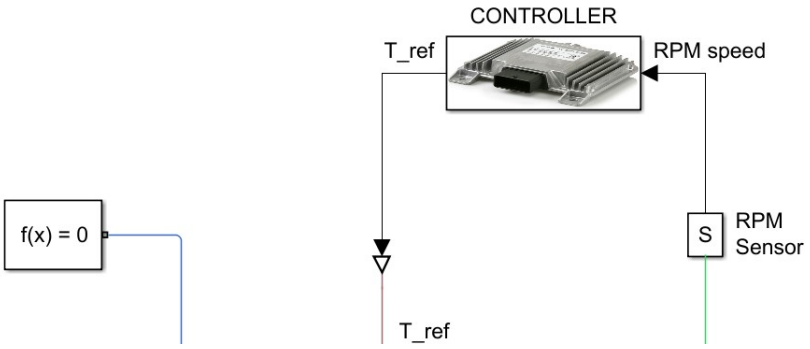

Figure 3.10: Electric Motor Controller Overview

Let's now move on describing how to model the electric motor in this BEV Simulink/Simscape model. As highlighted at the beginning of this section, many ways are possible to insert a working representation of the most important component inside an electric vehicle. During the thesis work, three different versions have been made, one of which has been considered the most efficient and complete. By the way the two remaining will be briefly explained since they required many working hours and it is also believed that they can be studied deeply during future thesis works and easily integrated in the already existing model. Electric motor can be modelled in 3 ways:

1. With its fundamental equations through SIMULINK.
2. With a PWM controller and a SIMSCAPE "DC motor" block
3. With a SIMSCAPE block called "Motor and Drive (System Level).

The third version has been considered the most effective and also the most similar to what has been previously modeled on OLT and for this reason it is the one implemented on the final BEV model and also the future HEV model.

With its fundamental equations through SIMULINK.

In the first version described the electric motor is not modelled as a physical component got with a block in Simscape library, but it is inserted in the model through its fundamental equations. The controller previously described in this section provides a physical signal T ref of the torque command to the electric motor. As shown in the fig. 3.11, this signal is sent as an input to a subsystem block, representing

the electric motor itself. This block needs to be linked with two reference ports, one electrical and one mechanical, which are respectively at 0 Volt and at rest. As output it has the port R which represents the transmission shaft of the vehicle (on which is also visible the RPM sensor previously described), and is also linked to the battery (not yet described) through an electrical connection. We can notice how Simscape makes interacting inside this component two physical domains, the blue electrical one and the green mechanic rotational one.

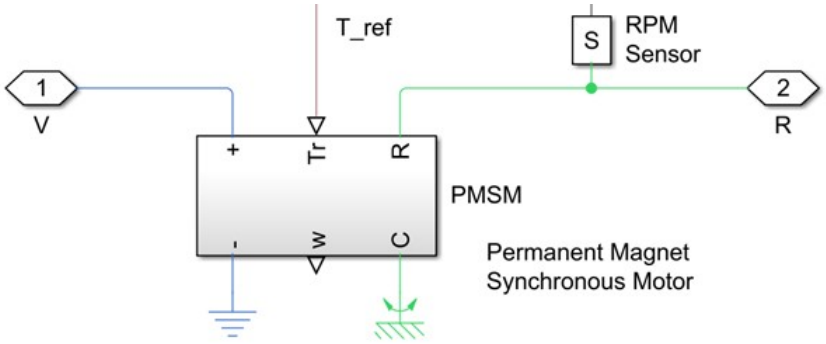

Figure 3.11: Electric Motor modelled through its Fundamental Equations

inside the PMSM subsytem, fig. 3.12 shows how other 3 subsystem are necessary in this case to model the speed torque interface, the fundamental motor equations and so losses, and the voltage current interface in the electrical branch.

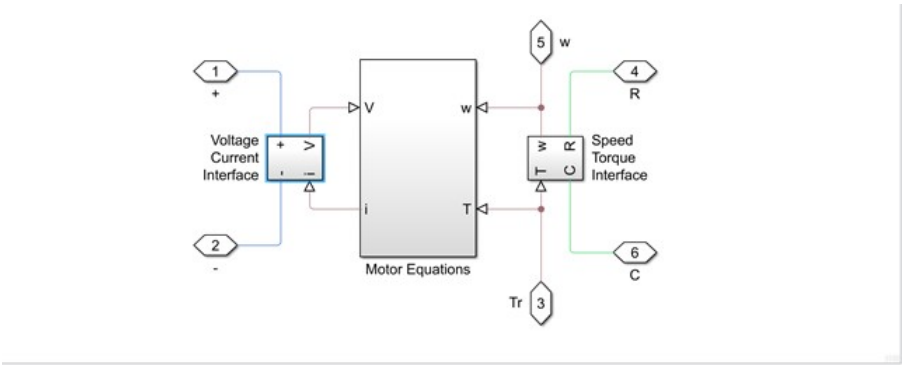

Figure 3.12: Electric Motor PMSM Subsystem

Inside the Motor Equations subystem, in fig.3.13 we can notice how the power and efficiency of the electric motor are modeled.

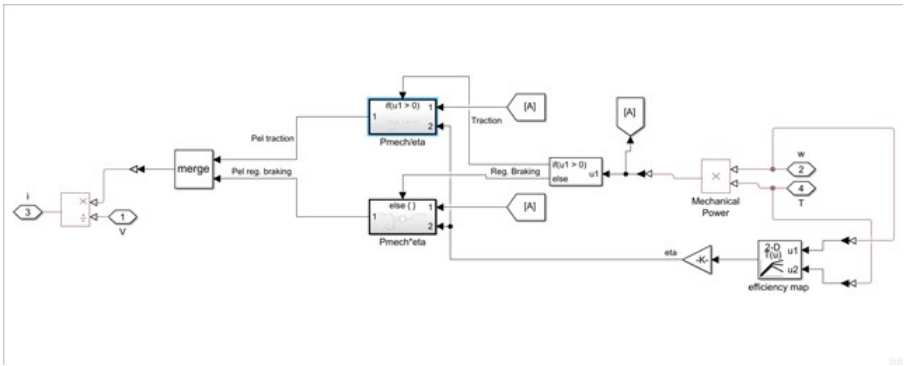

Figure 3.13: Electric Motor Equation Subsystem

The model considers electrical power required (or supplied) to battery as product (or ratio) between mechanical power (product between angular velocity and torque) and an efficiency factor which depends on an efficiency map of the electric motor, properly imported by OLT. To import this efficiency map on Simulink environment, a 2-D Look-Up Table block has been used. On datidelveicolo.m, it is mandatory to create

1. A vector N X 1 with omega values in rpm
2. A vector M X 1 with torque values on N*m
3. A matrix N X M with efficiency values (from 0 to 100).

Once the efficiency value has been computed depending on angular speed and torque values, it is necessary to impose an if-else logic. In fact computing electric power from the mechanical one is different if vehicle is requiring traction or in that moment it is regeneratively braking.

- If in fact omega and torque signs are the same, and so the mechanical power is positive, driver is asking traction to the electric motor, which means that the electric power is going to be computed as:

$$P_e = \frac{P_m}{\eta}$$

- If instead omega and torque signs are different, and so the mechanical power is negative, vehicle is recharging the battery, which means that electric power coming into the DC/DC converter (not yet described) is going to be computed as

$$P_e = P_m * \eta$$

To correctly impose the two situations it is as already said necessary an "IF block", which depending on P_m sign, addresses the P_m signals toward two "Action Block Subsystems" in which electrical power is computed with previously explained criterias. After that a merge block gathers electrical power value, which divided by voltage, gives our current output.

The just described solution revealed working in most of the simulation situations, but since no Simscape blocks have been used in this design phase and since the scheme is pretty complex, other solutions have been taken into considerations.

With a PWM controller and a SIMSCAPE "DC motor" block

Another possible solution considers the electric motor not commanded by the actuator designed so far, but by a PWM controller by means of a H-Bridge Driver. This model allows to by-pass controller design phase, using just Simscape blocks.

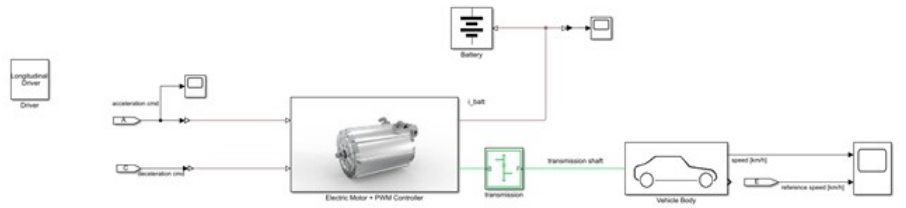

Figure 3.14: PWM Controlled Electric Motor BEV Model

Big modifications need to be applied to the entire scheme, which is shown in its final stage on fig. 3.14. In this scheme it is possible to notice the battery and driver blocks (not yet described), which feed a big block including the electric motor, drive and controller. Actually inside this latter also the battery is included, while the one shown in figure is just inserted as a "sample" in order to monitor charge and discharge phases. What is inside the electric motor block is shown in 3.15

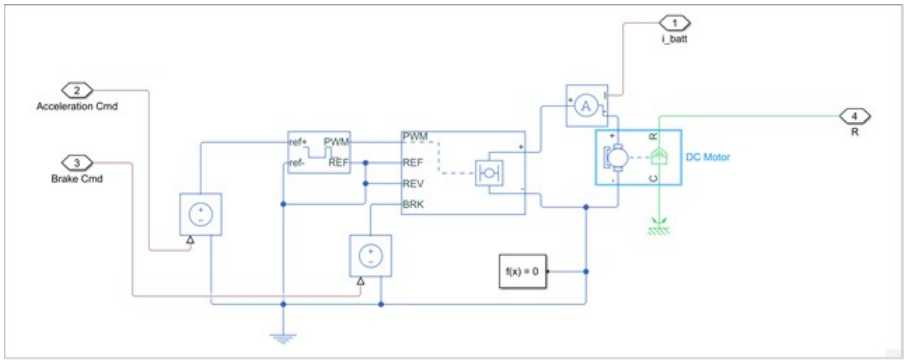

Figure 3.15: Electric Motor + Driver + PWM Controller Subsystem

Electric Motor is modeled as a DC motor, which terminals are linked to the corrispondent ones of a H-Bridge Simscape block. This latter receives an input from a PWM controller Simscape block, for the acceleration phases, and receives directly from the driver braking input. PWM itself receives input from the throttle pedal coming from the driver (converted in a current signal through an ideal current source block) . All the elements inside the blocks are properly parametrized, including the DC block in which an efficiency map coming from OLT is inserted. As previously already said, the model considers the battery inside the H-Bridge. "REV" terminal which represents the Reverse Gear motion is not taken into considerations since it is not required for OLT purposes. Also this alternative model showed to be stable in simulation phase, but there were some compatibility problem if referred to OLT model. In fact in OLT the PWM controller is not parametrized and the engine

or electric motors are controlled through a controller similar to the one previously designed, that in this model is not taken into consideration. Furthermore the PWM controller adds some parameter complication which are not easy to establish to be similar to the OLT design choice. This traduced to some differences in simulation results compared to the third model which will be discussed later on. In addition to this, this model considers the motor as a DC motor, which were practically abandoned in automotive industry and which can be responsible of some differences in results compared to the PMSM one modelled in other alternative models.

With a SIMSCAPE block called "Motor and Drive (System Level).

The third alternative to model the electric motor, which as already said is the one used in the next simulation showed and the one suggested to be used to OLT compatibility purposes, is the one depicted in fig. 3.16

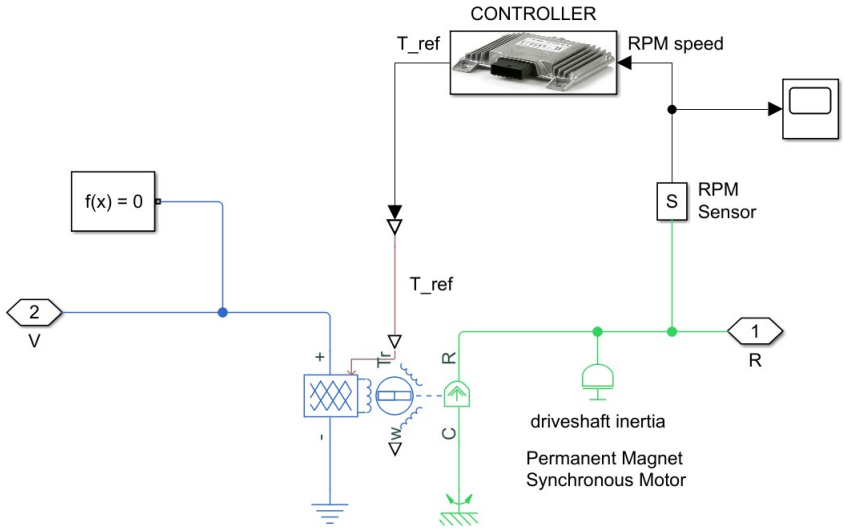

Figure 3.16: PMSM Electric Motor + Drive (System Level)

This model uses a Simscape block called "Motor and Drive (System Level)" which represents a generic electric motor mapped via limits curve and is able to autonomously compute the electric power sign inversion during the braking maneuvers. This block requires to be linked always to an electrical and mechanical reference, to the transmission, requires in input a T ref physical signal (like the one supplied by the controller) and requires to be linked to the positive pole of the battery (or the DC/DC converter). Block asks inside it to have uploaded max torque curve in function of angular speed and also the efficiency map of the electric motor in function of speed and torque. Both are extrapolated from OLT archives.

In addition to this, a simple inertia block is added to model the driveshaft inertia.

This model is by far the simplest one, but gives optimal and precise results in simulation phase. Furthermore it is a Simscape block, so it is perfectly fitting the aim of this thesis, and exploits commands coming from the controller already designed. It in addition needs exactly what we have in our OLT archive.

The model described so far is actually composed as depicted in fig.3.17

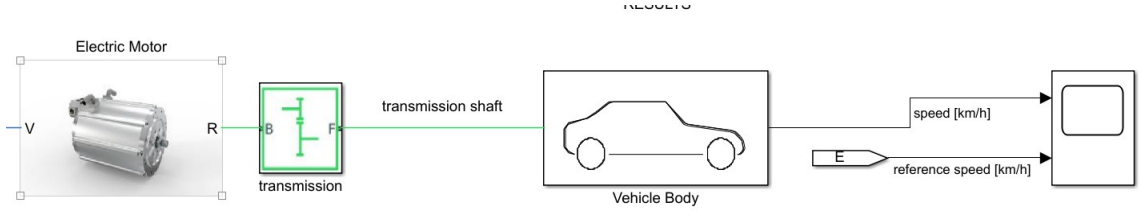

Figure 3.17: BEV model: Details of EM, Transmission, Body

3.2.4 DC/DC Converter

For what the DC/DC converter is concerned, this BEV model tries to be as similar as possible to the OLT model. In order to do so, the DC/DC converter is just modelled as an ideal transformer block which doesn't even alterate the voltage at its windings. It means that the winding ratio is always set to 1, as well as the efficiency. What is inside the DC/DC converter subsystem, which is linked by one side to the electric motor + controller subsystem, and by the other side to the battery subsystem, is shown in the next figure:

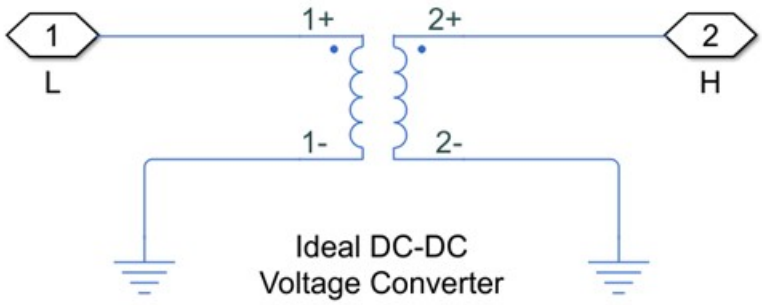

Figure 3.18: DC/DC Converter Subsystem

Note how it is fundamental to link the negative terminals of the ideal DC/DC converter to an electrical reference, set to 0 V. It is worth to point out that actually this component is not necessary for our simulation purposes, but is however modelled in order to be available in case of different purposes in next thesis works.

3.2.5 Li-Ion Battery

Battery subsystem is the most important inside a BEV model. In this case it is chosen to model a Lithium-Ion Battery, even if no specific assumption about the battery technology is made other than its Capacity, Equivalent Resistance and Open Circuit Voltage [**?**].
In very firsts phases of the model, the Simscape block "Generic Battery" has been used, but in order to be as similar as possible to OLT structure, a more complete

Simscape block is chosen. This block is present in the Simscape library and is labelled as "Battery (Table Based)". The scheme is shown in fig. 3.19.

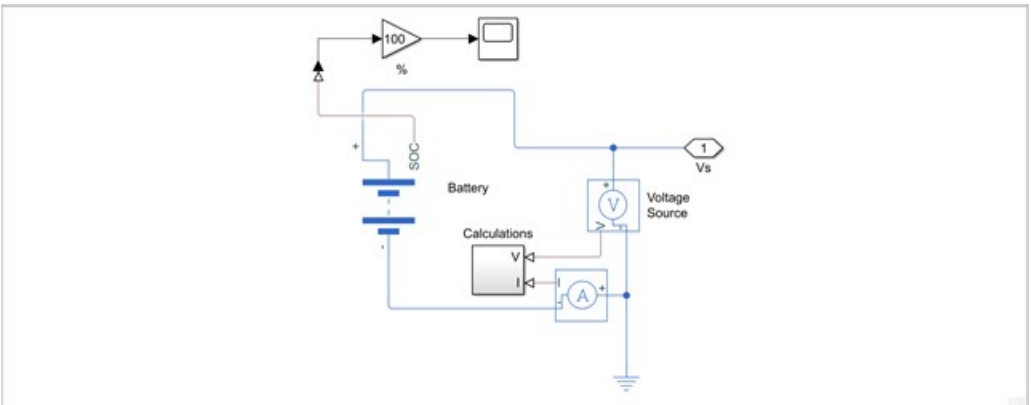

Figure 3.19: Li-Ion Battery Subsystem

Battery block must be connected to an electrical reference to its negative pole, to the DC/DC converter (electric motor) through its positive pole. Furthermore, the block requires in the parametrization phase to insert 3 vectors:

1. Vector of SOC

2. Vector of Equivalent Resistance of the battery dependant on SOC

3. Vector of Equivalent Open Circuit Voltage of the battery dependant on SOC.

In this way during simulations, Req and Vocv vary depending on the state of charge of the battery, which furthermore is automatically supplied as output variable by the battery block (as shown in upper part of fig.3.19) . These 3 vectors are extrapolated from OLT archives and correctly inserted in Simulink model via datidelveicolo.m.
No fade and no self-discharge phenomena are taken into considerations, as well as temperature dependant maps, since variables depend only on battery SOC.
In fig. 3.19 is also possible to note a Calculation block, which has been added in order to instantaneously monitor battery variables like Electric Power, Power losses because of Joule effect, Current and Voltage. This little subsystem is shown in fig. 3.20 .

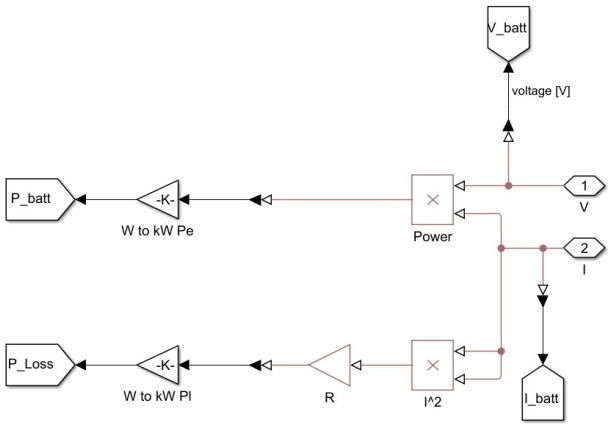

Figure 3.20: Li-Ion Battery Variable Calculation

These variables signals are sent through GoTo blocks towards a properly designed subsystem in order to show the results of simulations.

The actual model is shown in fig. 3.21.

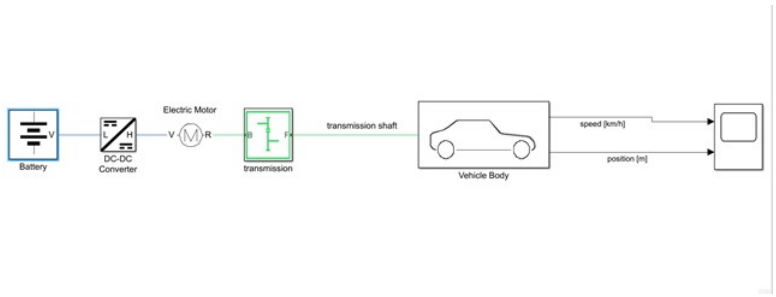

Figure 3.21: BEV model: Battery, DC/DC Converter, Electric Motor, Transmission, Vehicle Body.

What is now missing to BEV model is brake system and a longitudinal driver that actually follows a pre-established working cycle.

3.2.6 Longitudinal Driver and Brake Subsystem

Driver [10] has revealed to be one of the most affecting factor to the stability of the simulations. The choice is to model it as a PI controller, which instantaneously receives an input signal from a pre-established driving cycle, and, according to our forward-based model, depending on what is the actual vehicle speed compared to the target one, gives as output a throttle and a brake command signal, which are the ones already mentioned in electric motor/ generator + controller design phase. As already said, these two signals go from 0 to 1, are mutually exclusive and represent the percentage of pressing respectively throttle and brake pedals.

Driver is modelled through the Simulink block "Longitudinal Driver", which is the one suggested to interact with the "Longitudinal Body" vehicle block. A "Driver" subsystem is created, and what is inside it is shown on fig. 3.22.

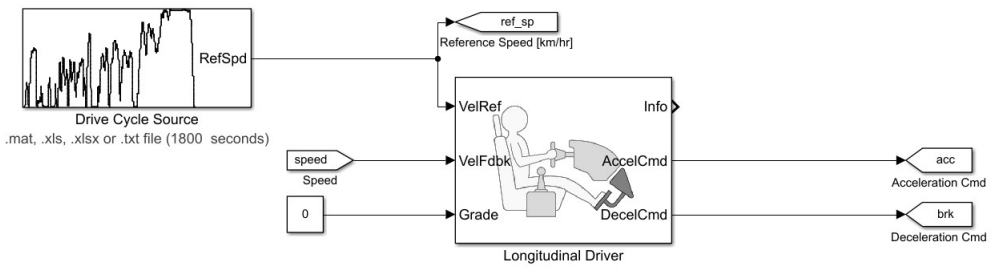

Figure 3.22: Longitudinal Driver Subsystem

This block requires in input some signals:

- VelRef: A drive cycle reference, imported from a proper block called "Drive Cycle Source", which contains most of the known driving cycles, but also the possibility to import an Excel file. This signal represents what the driver tries to follow during simulations.

- VelFdbk: Since we are dealing with forward simulation, this is the fundamental signal carrying on the actual vehicle speed.

- Grade: It represents the slope of the road that the vehicle needs to overcome. In this model we never deal with inclined roads.

While output of driver block are:

- AccelCmd: Already mentioned throttle command

- DecelCmd: Already mentioned brake command.

Parametrization phase is really a key factor for simulation stability. Driver blocks requires many PI controller parameters. These latter are very affecting factors, and a careful tuning phase has been necessary. At the end the parameters chosen are depicted in fig. 3.23.

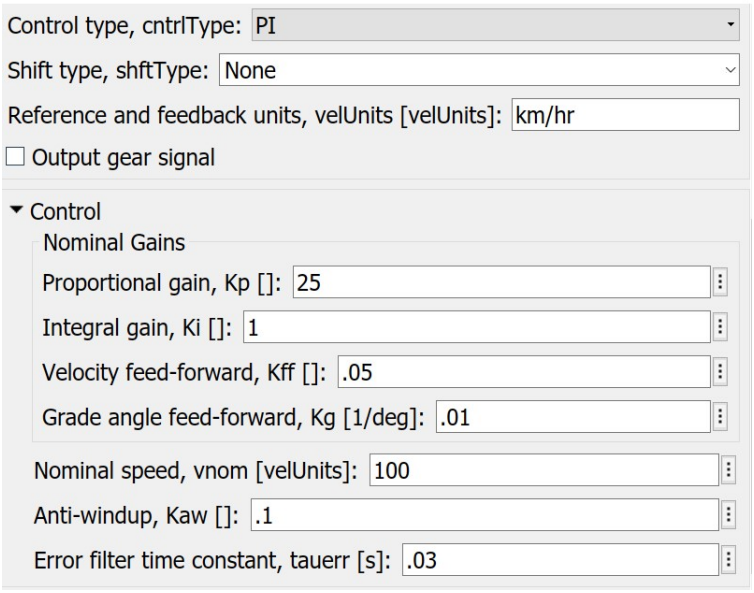

Figure 3.23: PI Controller Parameters

In particular the Proportional gain Kp, which can be seen as an "aggressiveness factor" of the driver, is set to a really high value. Despite this choice, the driver shows to be able to follow properly the driving cycle. This is probably due to the high versatility that the electric motor has. This factor will be strongly reduced in fact when a gearbox or an Internal Combustion Engine will be introduced, in conventional vehicle model.

For what the brake system is concerned,

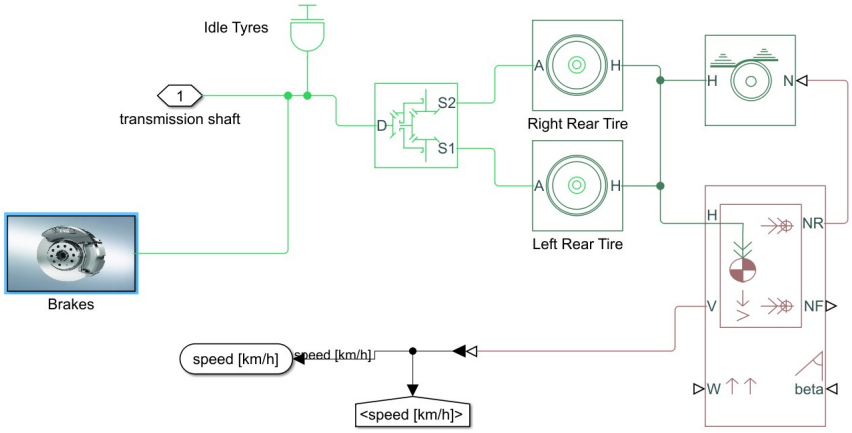

Figure 3.24: Brakes Subsystem Integrated in Vehicle Frame Subsystem.

brakes are simply represented as an Ideal Torque Source block, which is shown inside the Brakes subsystem, in fig. 3.25. This block needs to be linked to a mechanical rotational reference via port C, and gives to the port R (which is basically the semi-axle) the physical signal (converted by the PS-Simulink converter depicted) entering from port S. Port S is linked to a From block H coming from the Controller subsystem and carries on the Torque addressed to brakes as explained in previous chapters.

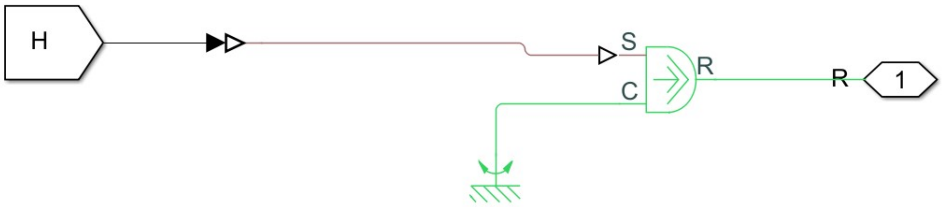

Figure 3.25: Brakes Subsystem Ideal Torque Source

3.2.7 Final Model Appearance

Now that all of the BEV model components have been analysed in details, a picture of the complete and definitive model is shown:

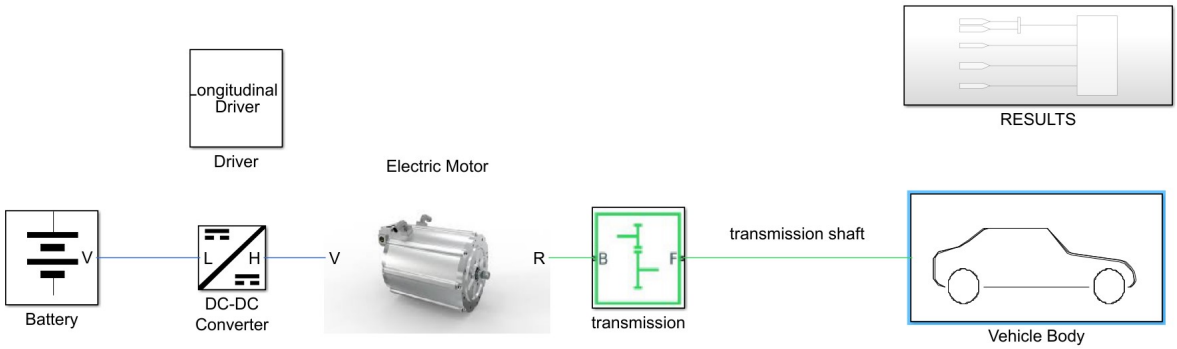

Figure 3.26: BEV Definitive Model

It is evident the series of components, starting from the battery which is clearly the power source, passing from the DC/DC Converter and entering into the Electric Motor block, from which a mechanical rotational shaft exits, passing from the Final Drive, directly toward the Vehicle Body Block. A " Results " Subsystem is alto present to look at results after running simulations.

3.3 Simulation

A final simulation of the model described so far is run. This simulation is not supposed to show results similar to OLT model, since neither a heavy duty vehicle is updated, but is just executed in order to show the compatibility between elements designed in the model. More details are provided in chapter dedicated to comparison between SIMSCAPE and GT-DRIVE model.

Parameters used to characterize the model in this simulation are shown in fig. 3.27

Parameter	Value	UM
Driving Cycle	WHDC	[-]
Mass	3500	[kg]
Wheels per axles	2	[-]
Cog-Front wheel dist	1.5	[m]
Cog-Rear wheel dist	1.5	[m]
Cog Height from ground	0.55	[m]
Frontal Area	4.5	[m²]
Cx	0.31	[-]
Rolling Resistance Coeff.	0.00896	[-]
Driveshaft Inertia	0.32	[kg×m²]
Rolling Radius	0.2032	[m]
Wheel Inertia	2	[kg×m²]
τ Final Drive	12	[-]
η Final Drive	0.98	[-]
Electric Motor Inertia	0.15	[kg×m²]
Battery Vocv (100% SOC)	335	[V]
Battery Req (100%SOC)	0.1525	[Ω]
Battery Ctot	58	[A/hr]

Figure 3.27: Simulation Parameters Table

Results are shown in the next images:

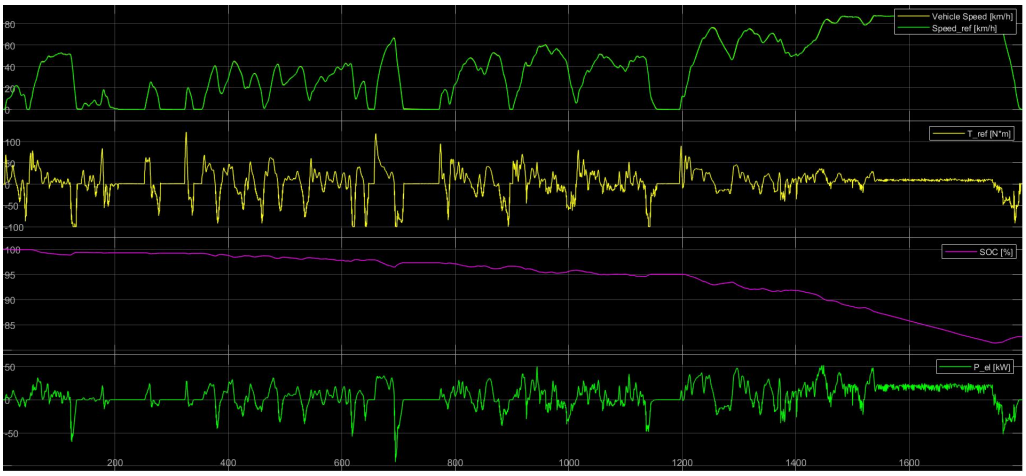

Figure 3.28: Simulation Vehicle Results: Vehicle Speed, EM Torque, SOC, Pel

In fig. 3.28 Vehicle Speed comparison with Reference Speed, Electric motor torque, SOC of the battery, and Electric Power are shown. It easy to note how the driver is able to closely follow the driving cycle, since the two curve are quite superimposed. About torque command to the electric motor, it is worth to note how torque inverts its sign during the braking maneuvers, and also how braking logic actuated by controller makes the electric motor never overcome threshold value of 100 N*m. In fig.3.29, which shows the braking torque splitting, it is evident how the controller works. The brakes are involved only when necessary.

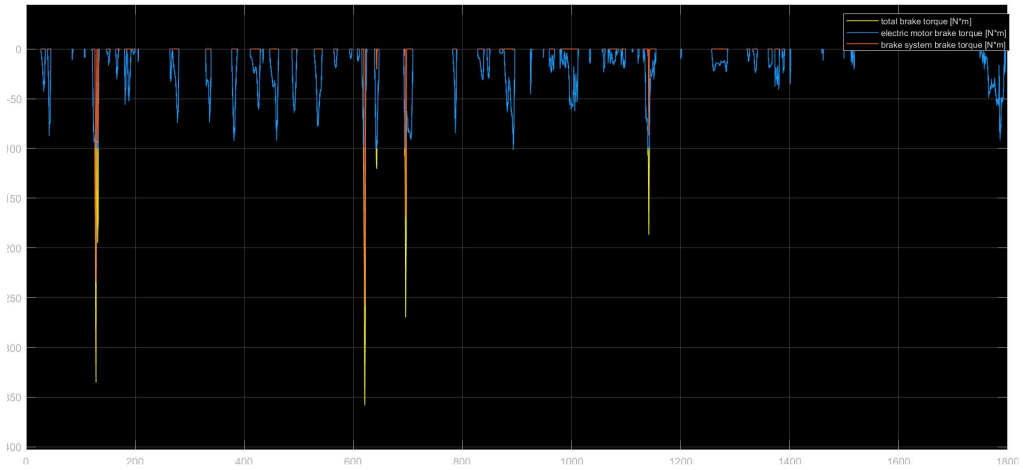

Figure 3.29: Braking Torque Ripartition

About State of Charge of the battery, we can notice how starting from a 100 value it goes discharging during the cycle, even if it is correctly recharged during the braking phases. This phenomena is confirmed from the inversion in sign of the electric power, which indicates the phase in which the battery is recharged.

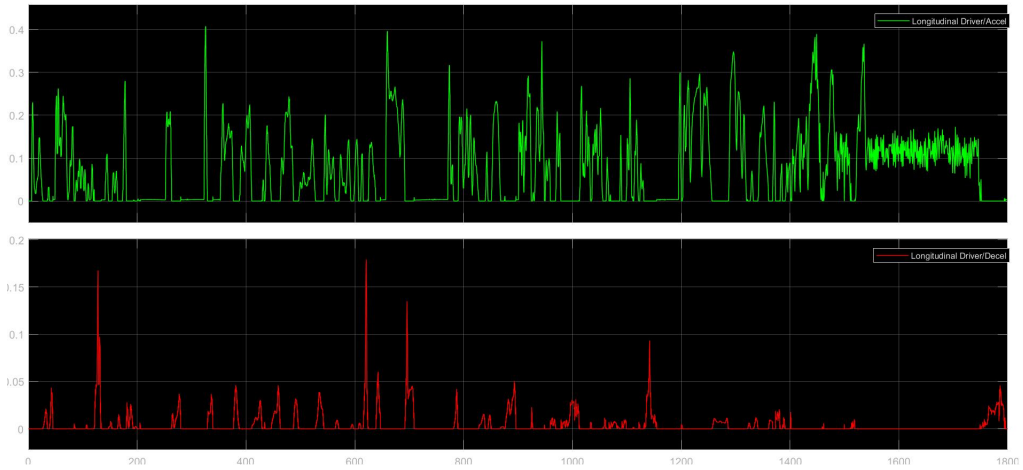

Figure 3.30: Pedal Commands

Fig. 3.30 shows instead the driver throttle and brake pedal commands. This image shows how the driver acts during the driving cycle path. It is also evident how the two signal are mutually exclusive.

Chapter 4

Conventional Vehicle

Main aim of this chapter is the design of a Conventional Vehicle model on Simulink /Simscape platform. Blocks are designed as done for the BEV project, with similarities and differences clearly due to the different adaptation of components and type of propulsion used. By the way the basic idea is to use as much as possible component so far designed in battery electric vehicle, as well as keeping very close to idea adopted in OLT model. Clearly some Conventional Vehicle components are strongly different with respect to their corresponding BEV ones. In this cases, different philosophies have been adopted. Another matlab file "datidelveicolo.m" is created and by means of it is possible to insert parameter of the model.

4.1 Model Creation

Model is created with same procedure applied for the BEV. Another Simulink document is created and again subsystems representing the main component of the Conventional Vehicle are created. They are deeply analysed in this chapter. Because of the complexity of the design procedure, especially for what gearbox is concerned, in this case a gradual descriptive approach is adopted, showing step-by-step choices taken during the model building.

4.1.1 Vehicle Body and Driver

As previously said, when possible, the idea is to use again same component adopted in BEV model. In the case of Vehicle Body and Driver for example, subsystem are the same because there are no significant differences during the design phases, since aerodynamic and cooling factors are not taken into account in this model. For this reason, same blocks are adopted, and same GoTo and From blocks are used. Some modifications are applied to Longitudinal Driver block parametrization phase. An iterative procedure has been executed to find right driver aggressiveness factors. At the end fig.4.1 shows the combination of parameters that best fits the simulation.

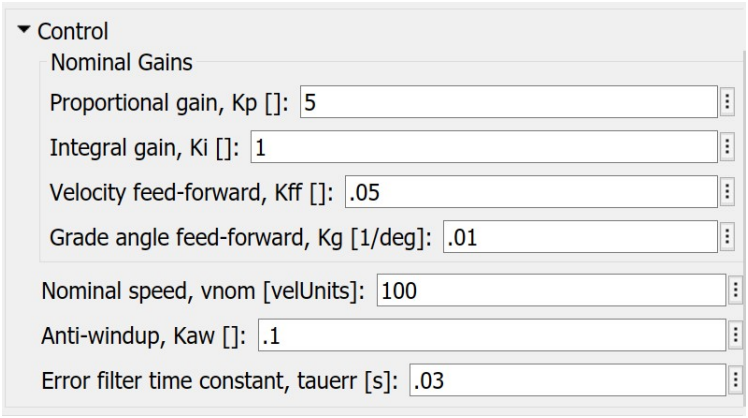

Figure 4.1: Conventional Vehicle Driver Parameters

4.1.2 Transmission

Transmission subsystem is by far the most complex one to be designed. In fact it is completely different from what in BEV is designed, because conventional vehicles need a gearbox which is in charge to adapt torque and speed required from ICE towards the driveline. What was called in BEV model "transmission", here represents only the Final Drive, which clearly is not a strong value as it was in the previous model, because now we have to split the total reduction in 2 components: one represented by the final drive, one represented by gearbox. Inside our CV a 6-speeds gearbox must be properly inserted. Gearbox parameter are taken from OLT database, and fig.4.2 shows a little Excel table got from vehicleData.xls. These data are thought to be adopted for a heavy duty vehicle, but the gearbox consists of a 6-speed transmission manual device that perfectly fits this thesis purposes. For this reasons, same gearbox parameter are adopted for the CV model.

GEARBOX PARAMETERS						
Gear number	1	2	3	4	5	
Gear ratios [-]	6.02	3.32	2.07	1.4	1	0.7
Gear efficiency [-]	0.9675	0.9673	0.9685	0.971	0.99	0.9
Gearbox inertia coefficient [kg*m^2]	0.15					
Outer (wheel side) gearbox inertia coefficient [kg*m^2]	0.05					
Inner (pwt side) gearbox inertia coefficient [kg*m^2]	0.05					
Gearbox mass [kg]	100					

Figure 4.2: OLT Gearbox Parameters

First thing to design is inertia of gearbox itself, outer part of the gearbox and inner part of the gearbox. 3 Inertia blocks are properly inserted in the model, one upstream of the gearbox, one inside, one downstream. For the values, they are taken from matlab file "datidelveicolo".
To sum up, Transmission Subsystem is by now composed as shown in fig. 4.3. It is possible to distinguish in this picture the Gearbox and the Final Drive, which is schematized as in BEV model as a simple gear pair.

Inside the "Gearbox" subsystem, we are going to design the layout of our 6 speeds gearbox. The first thing we have to insert is a device which is in charge to simulate the friction clutch commanded by the driver if he upshifts/downshifts. The box chosen for this purpose is "disk friction clutch". It models a dry clutch. For what parameter data are concerned, the ones preset by Simulink are for this thesis

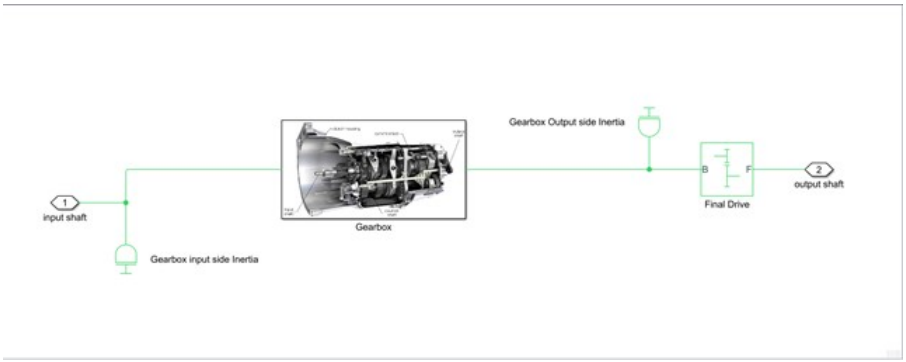

Figure 4.3: Transmission Subsystem: Gearbox + Final Drive

purposes enough [11], since we don't have specific information on our OLT archive. Block Disk Friction Clutch presents 3 ports, 2 to the left, 1 to the right.

- P: to the left, it indicates the physical signal value in input to the clutch. If P is bigger than a P threshold (100 Pa), the clutch is engaged, and the physical connection between input and outport ports B and F is realized. It must be linked to an actuator block, which gives the signal for each gear to be selected.
- B: to the left, represents the input port. It is linked to the engine part of the gearbox.
- F: to the right, represents the output port. It is linked to the wheel part of the gearbox.

The idea is to create six in parallel lines, one for each gear, which will be one at time activated and which are "run across" by power, depending on the gear selected by the driver.

As said before, the port P of the clutch has to be fed with the command coming from an actuator. This little subsystem is shown in fig. 4.4.

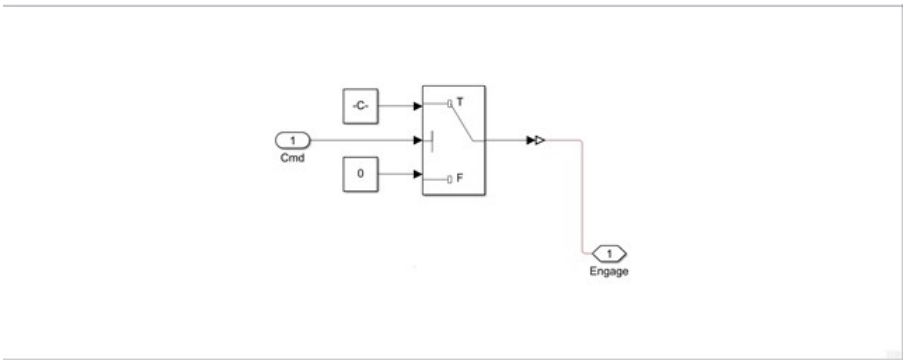

Figure 4.4: Clutch Actuator Subsystem

It has in its input a command signal, which will be properly designed later on and will carry on just a signal bigger than 0 when the gear will be selected by the driver. Entering in a switch, this signal produces a pressure signal which activates the clutch, so to engage it. This value of pressure must be sufficiently high to make the engagement of the clutch smooth and fast. For our purposes we choose a value of

$1e9$ Pa. What remains to be designed now for the first gear is just the gear meshing pair, which will be modelled as a "simple gear" block. The gear ratio will be the one of our OLT gearbox, as well as the efficiency of the gear mesh. They are both imported from matlab.

Our "one gear gearbox" is so far composed as shown in fig. 4.5.

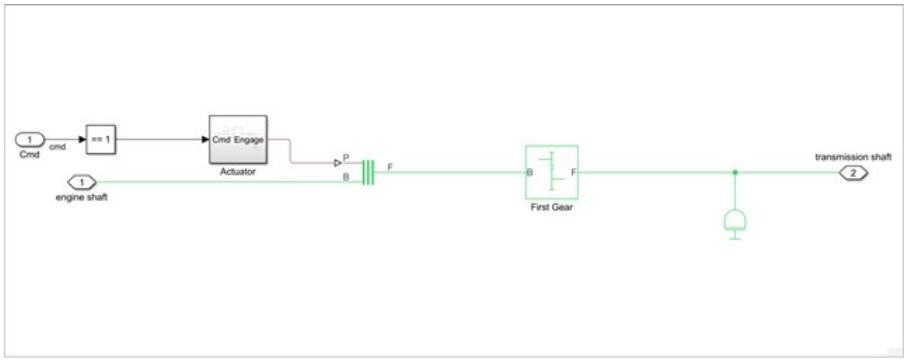

Figure 4.5: One-Speed Gearbox

What we have to do now is just copy and paste this layout six times, inserting always a different gear mesh, with different gear ratios and efficiency, gear for gear.

Adding other gears and an inertia block for each mesh which simulates the inertia of the gears, we obtain what is shown on fig.4.6.

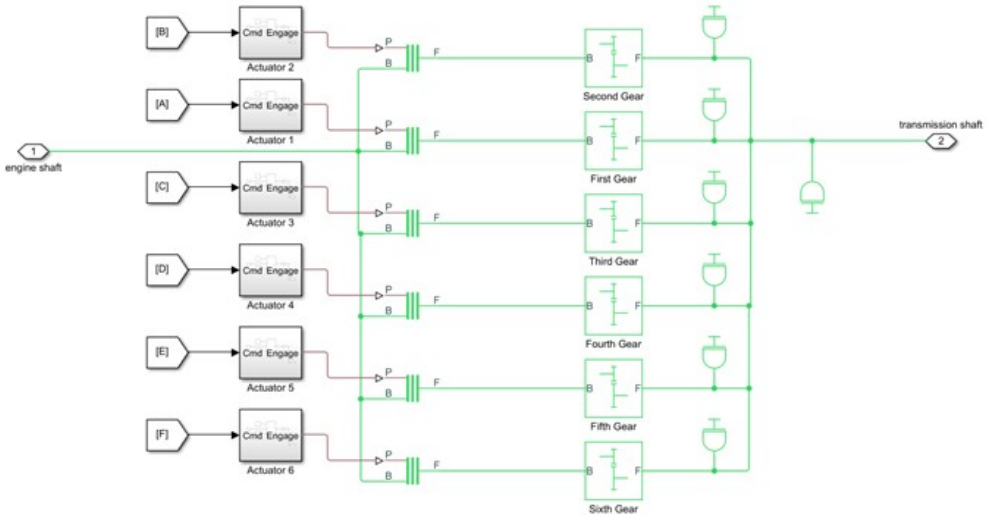

Figure 4.6: Six-Speeds Gearbox

Note how each gear is properly linked with a dedicated actuator, which by now is not fed by anything, because the logic control strategy for the gear selection will be later on designed. The B port of the clutch is always connected to a node coming from the engine shaft and the F port of the gear is connected to another node in connection with the driveline. In this way we will "use" always one path passing always from one gear mesh, starting from the same point (engine shaft), going always to the same point (transmission shaft). What now remains to be designed is the feed port of the actuators. We will use for this aim some GoTo blocks, arranged as shown on fig. 4.7

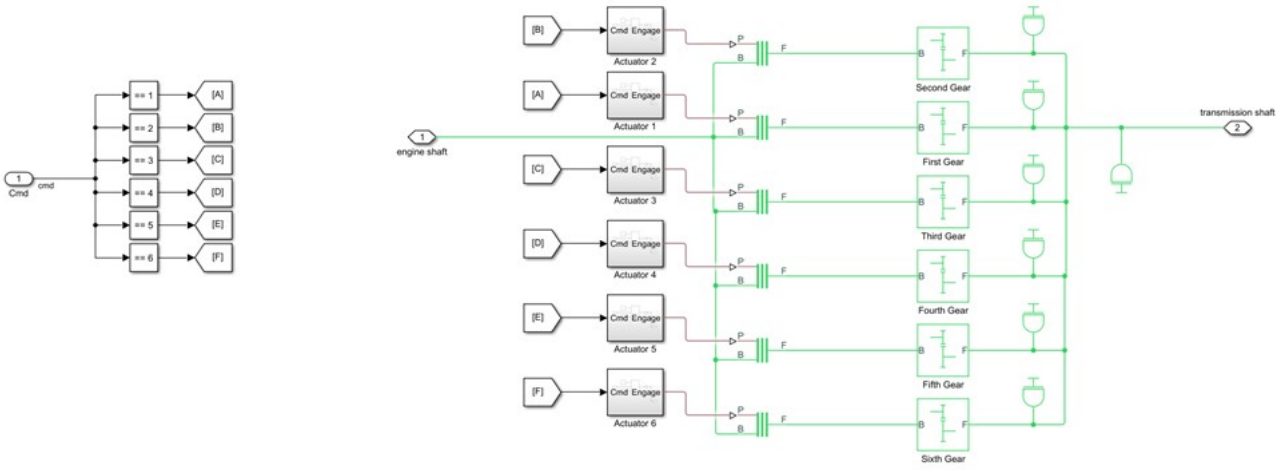

Figure 4.7: Six-Speeds Gearbox + Commands Signals

On left hand side we can notice the cmd signal (yet not designed), which will be addressed to the A, B, C, D, E, F GoTo block, depending on if it is 1, 2, 3, 4, 5, 6 gear signal. Each go to block will be addressed to the actuator, which feeling the signal will engage the clutch and so the right gear. What now is important to do is designing an internal combustion engine which can provide power and torque to the gearbox.

4.1.3 Internal Combustion Engine

[12] Before designing the upshift and downshift logic controller, what we are going to approach is the engine design, so to have something which can give power to our system. What we want to insert in our model is another subsystem called "ICE". This block doesn't have a feeding port (as it was for the electric motor in the BEV

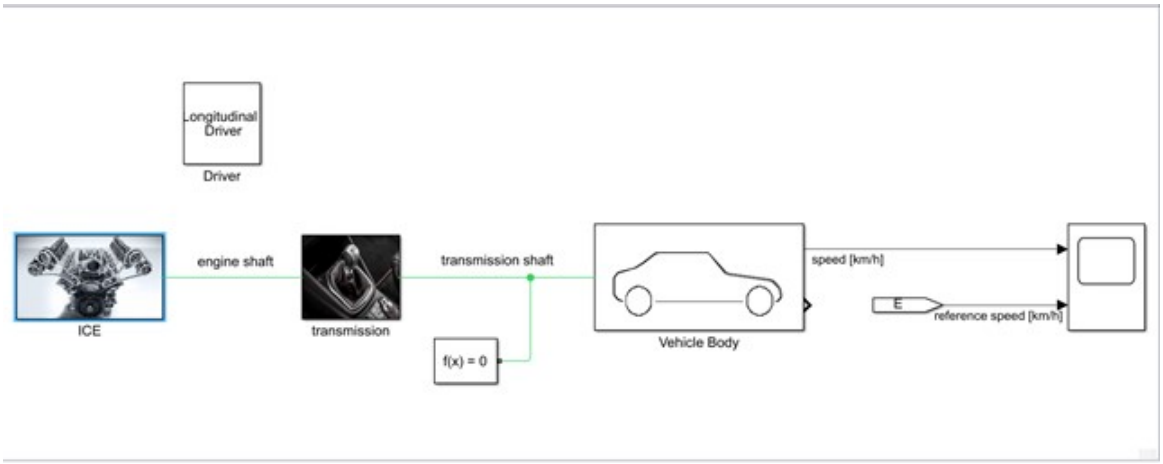

Figure 4.8: Conventional Vehicle: ICE + Transmission + Vehicle Body + Longitudinal Driver Blocks

model), because we get our energy from the combustion chamber inside it. What we have is instead an outport port linked to the transmission, which is our engine crankshaft. Inside the block "ICE" we choose to model the engine with the simscape

block “Generic Engine”.
Fig. 4.9 shows what is inside the ICE subsystem.

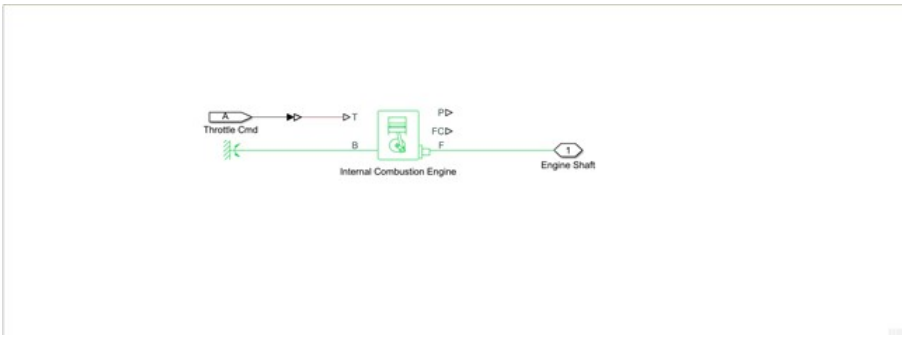

Figure 4.9: Internal Combustion Engine Subsystem

This fundamental block has several inport and outport:

- T: input of throttle position (from 0 to 1) coming from the driver output signal.
- B: input of the rotational reference for the engine, which in our case is a “mechanical rotational reference” block, since we are moving with respect to the ground.
- P: power output port, which for the moment will not be utilized.
- FC: Fuel consumption output port, which can be used for fuel optimization.
- F: output port linked to the crankshaft of the engine, which will gather power from the engine itself.

What now is mandatory is correctly parameterize the block ICE inserting the correct data of our engine. Many setting parameters are required:

1. Engine torque: For what the engine torque is concerned, we choose to model the limit curve of the engine with the option “Tabulated Power Data”, since we have in our OLT archive data of maximum power developed by the engine in function of speed[**?**]. So we import from excel file to datidelveicolo.m vectors of speed (in rpm) and corresponding maximum power (in kW), which will feed the Simulink file.
2. Fuel Consumption: FC is evaluated with the option “Fuel Consumption by speed and torque”. It is mandatory to insert a map of brake specific fuel consumpion or fuel consumption of our engine, always taken in our case from OLT archive [**?**]. Data are imported on “datidelveicolo.m” coming from a matlab code which is in charge to create the right map.
3. Dynamics: the parameter to be specified in this case is the speed threshold, which represents the width of the speed range over which the engine torque is blended to zero as omega approaches the stall speed. In our case it is selected 750 rpm as this threshold value.

4. Idle and Red Line Controllers: the ICE model contains a specific section in which it is possible to create controllers of Idle Speed and Maximum Speed. The first one gives a throttle command, also if not specified by the driver, in case in which the engine is falling down the idle speed. The second one limits instead the maximum speed that the engine is capable to tolerate. Also in this case a tuning process is necessary to set the right value of aggressiveness of this specific controller, in order to avoid excessive throttle actuation or on the other side that the engine speed falls anyway down the idle threshold. [12]

In particular for the latter point, final tuning parameters are shown in fig. 4.10.

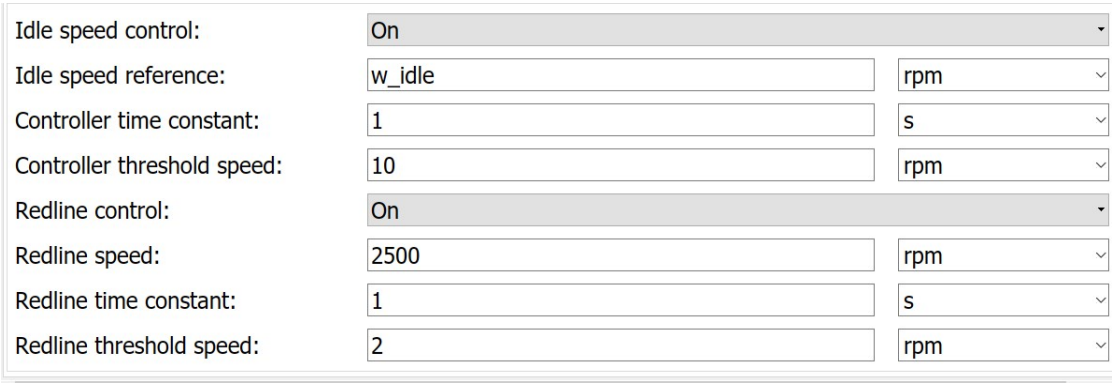

Figure 4.10: Idle Speed and Redline Controllers Parameters

About the idle controller, the speed below which the engine must not fall down is set to the idle one, so 750 rpm. The controller time constant is a parameter which indicates after how many seconds the controller has to intervene when the engine speed is approaching to the idle speed reference one, and the preset value is chosen. Key parameter is instead the Controller Threshold Speed, which can be seen as an "aggressiveness factor" of the controller . In fact, large values decrease controller responsiveness, while small values increase computational cost.
Same considerations apply for the RedLine controller.

4.1.4 Brake System

For what the brake system is concerned, some modifications are applied with respect to the electric vehicle model. This is due to the fact that the braking logic in this latter splits the torque to brake the car in two contributions, while in this case, since there is no battery and no regenerative braking, all the braking torque will be addressed to the brake system. Clearly no regenerative braking will be performed.

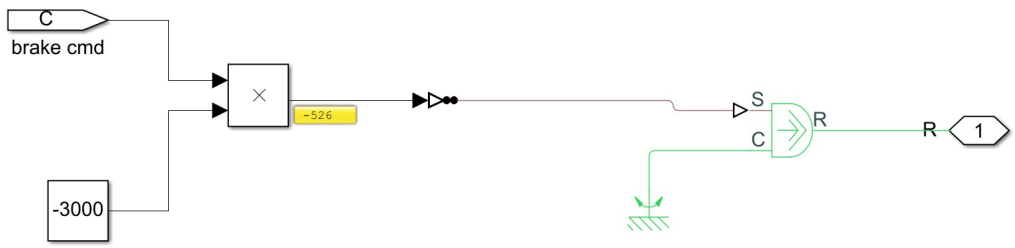

Figure 4.11: Brake System in Conventional Vehicle Model

How fig. 4.11 shows, in this case the brakes are modeled as an ideal torque source, as in the BEV model, but the torque signal is just a matter of multiplication between a maximum torque value, (-3000 N*m) and directly the brake pedal command coming from the driver subsystem, without any controller or any braking logic.

4.1.5 Upshifting / Downshifting Logic Controller

Last step before running the simulation is designing a solid gearshift logic control, which allows our driver to exploit the transmission to modulate the torque required to our engine. This logic is not easy to implement on Simulink environment. The idea is to develop a gearshift logic based on rpm speed of the engine.
When speed falls below a threshold value the driver should downshift; when speed overtakes a max threshold, the driver should upshift; when the engine is in between these two thresholds, the driver mustn't change gear.
The blocks "if statement" with their related "action subsystem" are too much complicated for this purpose, since too many loops should be created, and the stability of the entire system would be compromised.
The tool chosen to implement this kind of logic is the Simulink add-on "Stateflow", which is basically used when this kind of decisional logics have to be taken by the software. What we need is a "stateflow chart", a block that allows autonomously to make a transition from a "state" to another one, basing on some criteria and input we are going to design. In our case:

- States will be gears, for example first, second, and so on
- Criteria will be the one already explained of speed threshold for the moment. When the engine speed exceeds a certain threshold, the stateflow chart must autonomously understand that the gear has to be changed.
- Input will be clearly the engine speed, so that to allow the stateflow chart operator to make the transitions at the right moment. Same thing but opposite states for downshift procedures. Another input will be the reference speed in km/hr, which will be used as explained later on only passing from neutral to first gear.

- Output of the chart must be the gear, which will enter into the gearbox block as input command, selecting the gear as previously explained in the design phase of the transmission.

We start creating the block "chart" from Simscape library [13], in which the rpm speed coming from a rpm speed sensor (as in BEV model) and the reference speed are inputs, and which will give as output a signal y which represents the gear selected and is just a numeric signal going from 0 (neutral) to 6. This block is called "gearshift logic"

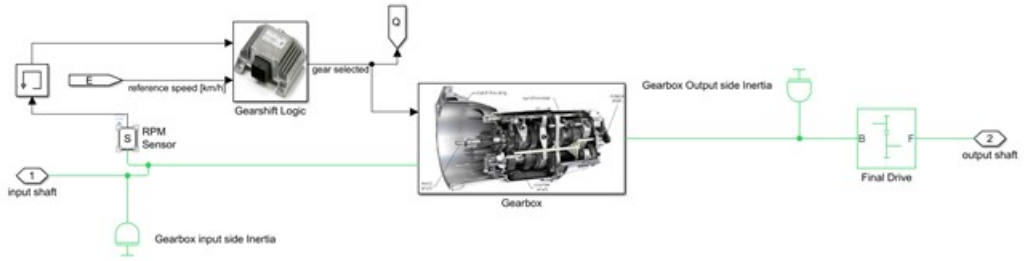

Figure 4.12: Transmission Subsystem + Gearshift Logic Controller

As we can see in this case the memory block is necessary to guarantee the stability of the system.
Inside the block "gearshift logic", we select the block "state". Each state is a particular condition, which if verified, gives a particular output at the block "gearshift logic". For example, let's examine a part of that in fig. 4.13

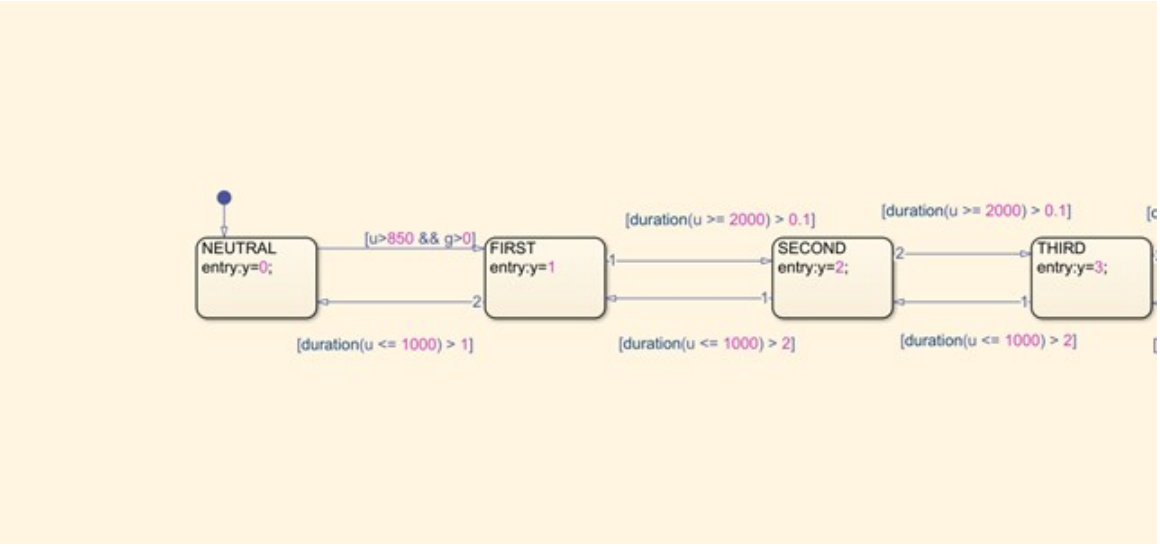

Figure 4.13: View of StateFlow Gearshift Controller

The first state (the one at left hand side), represents the situation called "Neutral", and if verified, it gives as y output signal of the "gearshift logic" y = 0. This

means that whenever the engine is in neutral, the input of the gearbox will be "0", so no clutches will be triggered, so no power transmission toward the driveline will be present. Inside the gearshift logic, neutral is always the initial condition. It means that if we basically start a simulation, the engine will be immediately on, but the very first iterations will give 0 as input for the gearbox.
The second state, the one labelled as "first", indicates the case in which the driver selects the first gear; in fact it will give as answer y=1, so the gearbox will immediately engage the clutch 1, associated with the first speed gear pair and so on. What basically is missing is to impose the condition for which we should pass from neutral to first gear. To do so, we use the arrow depicted in fig. 4.13 linking NEUTRAL state and FIRST state. What is written inside the square brackets is just the condition that has to be verified in order to allow a transition in the sense indicated by the arrow tip. For example, for the moment, we impose that the driver will engage the first gear coming from neutral only when the engine will be at 850 rpm and at the same time the reference speed (labelled as g) is bigger than 0. The reason why we have to impose also the second condition is to avoid that the driver engaged the first gear not properly, in situation in which the engine speed is for example higher than the threshold but the car should not move.
In this way for example at the very first stages of the cycle, the engine will be at idle speed. When after some seconds the car should start to move, the reference speed will be bigger than zero, so the longitudinal driver will push the accelerator pedal and the engine speed will overcome the 850 rpm threshold. At that point the first gear will be engaged and the motion will start.
At the same way, we impose that the driver should pass from first to neutral when the engine speed falls below 1000 rpm for more than 1 seconds. This duration function is necessary, because sometimes, passing from neutral to first gear, the engine speed falls temporarily down the threshold of 750. This is due to the inertia of the vehicle. If we didn't impose this condition, the gearshift logic would engage again the neutral gear, hindering the movement of the vehicle. This time window of 1 second is for now just a symbolic value, which will be optimized later on when the simulation will be run.
Another block is created, which represents the second gear engagement. Criterias for which driver passes from first to second gear are different from the ones previously explained between N and 1°. The driver will upshift as soon as the engine reaches 2000 rpm for more than 1 ten of second. This choice has been arbitrarily taken in order to model normal human reaction times. The driver will instead upshift if engine speed falls down 1000 for more than 2 seconds. The 2 second threshold is imposed for the same reason as before. Clearly the logic will be repeated for other gears up to the sixth, which is not depicted in figure because not yet designed.
Let's imagine to have a 3 speed gearbox instead of our manual 6 speeds transmission. Let's run a simulation to just gather what has been done so far and look at the results. For the moment we will just impose as drive cycle source an acceleration phase, a coast phase, and a deceleration phase as shown in the fig. 4.14.

Figure 4.14: Reference Driving Cycle for a Basic Simulation

A simulation with plausible vehicle data is run, just for reference and with our very rough gearbox. Results got from the vehicle speed scope are plot in fig. 4.15

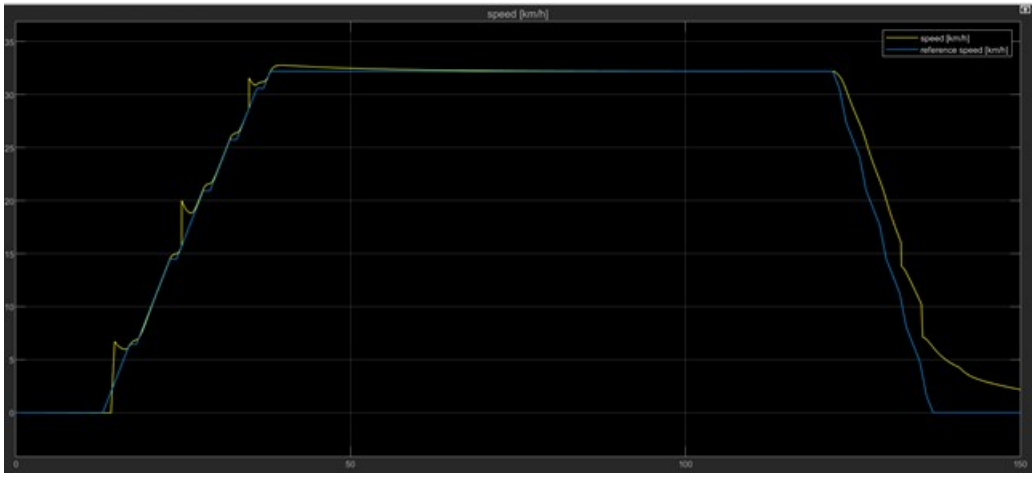

Figure 4.15: Results of Basic Simulation: Vehicle Speed

It is also chosen to plot the speed of the vehicle (below), with the omega engine (in the middle) and the gear selected (upon): at the lower position we also plot accelerator and braking commands. Results are shown in fig. 4.16

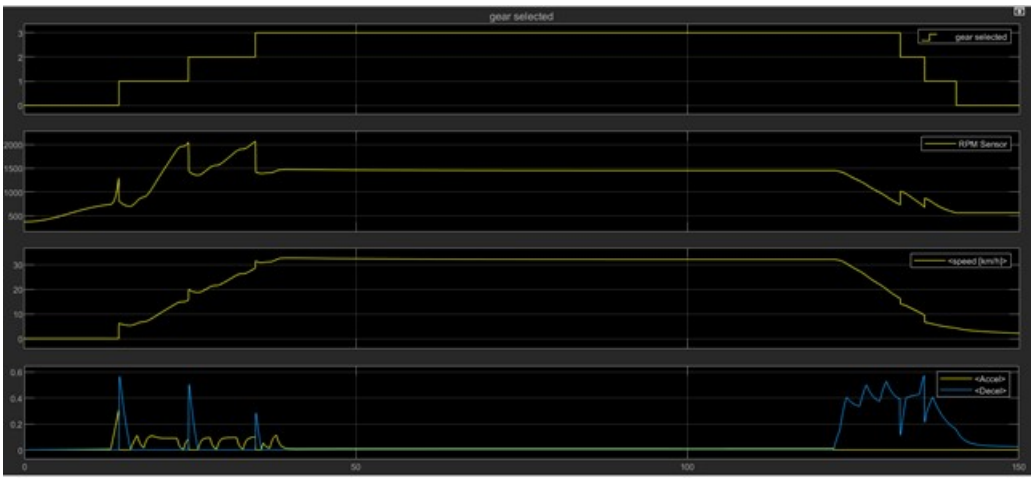

Figure 4.16: Results of Basic Simulation: from top to bottom: Gear, Engine Speed, Vehicle Speed, Driver Commands

Looking at the image we can notice how after some seconds the accelerator is pushed by the driver. Immediately omega engine increases, up to the 850 rpm threshold, after which the first gear is engaged. We can also notice how the driver is forced to brake heavily after having selected the first gear, because the engine speed is higher than advisable, and he tries to report it back to the right value. Probably all the parameter that characterize the gear logic selection should be optimized, but the system works.

What is now missing is to implement the gearshift logic for the remaining gears, as shown in 4.17.

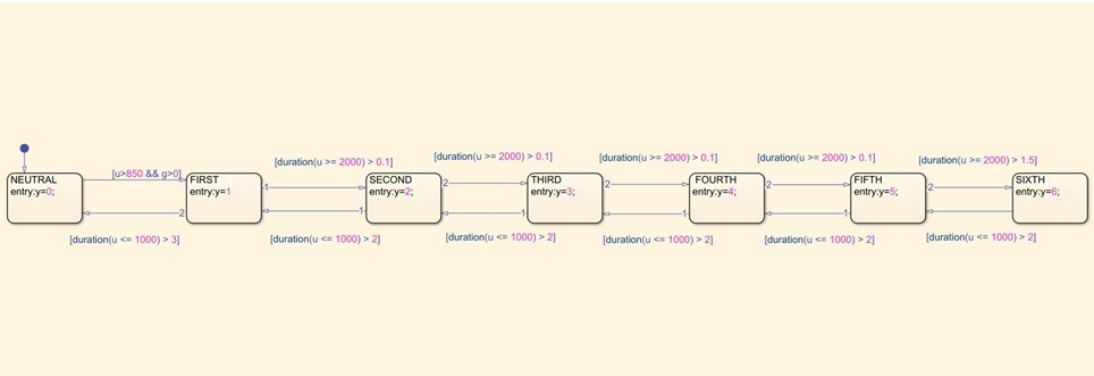

Figure 4.17: Six-Speeds Gearbox Gearshift Logic

Let's now run a simulation pushing a little bit more our vehicle, in order to stimulate the driver to select also higher gears

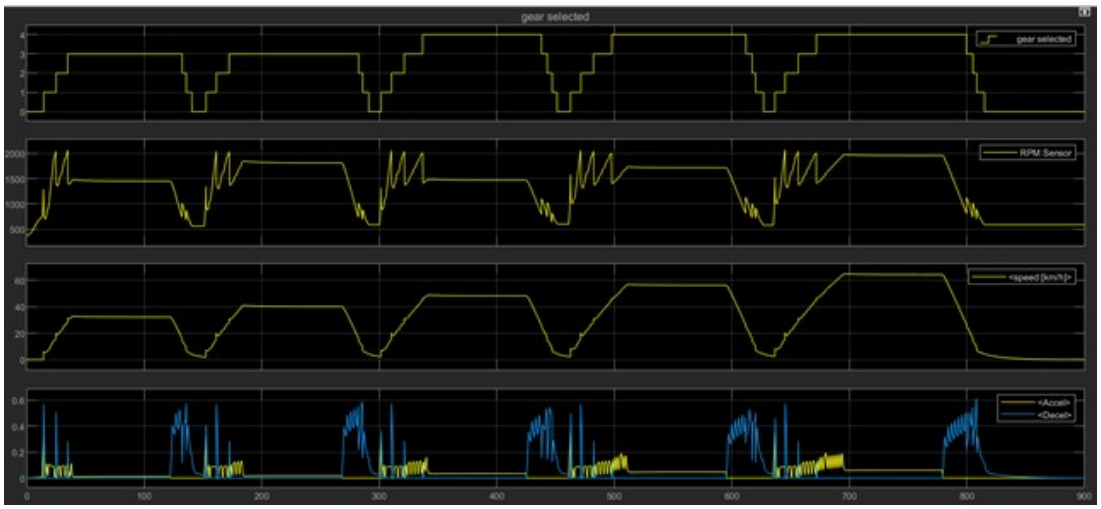

Figure 4.18: Results of Simulation 2

Our cycle is a little bit more requiring in terms of max speed. In fact the driver selects the 4° gear, and in the last max speed section is also near to reach the threshold of the 2000 rpm to select the 5° one. Let's diminish the threshold value for passing from 4th to 5th gear to 1800, and let's run again the simulation.

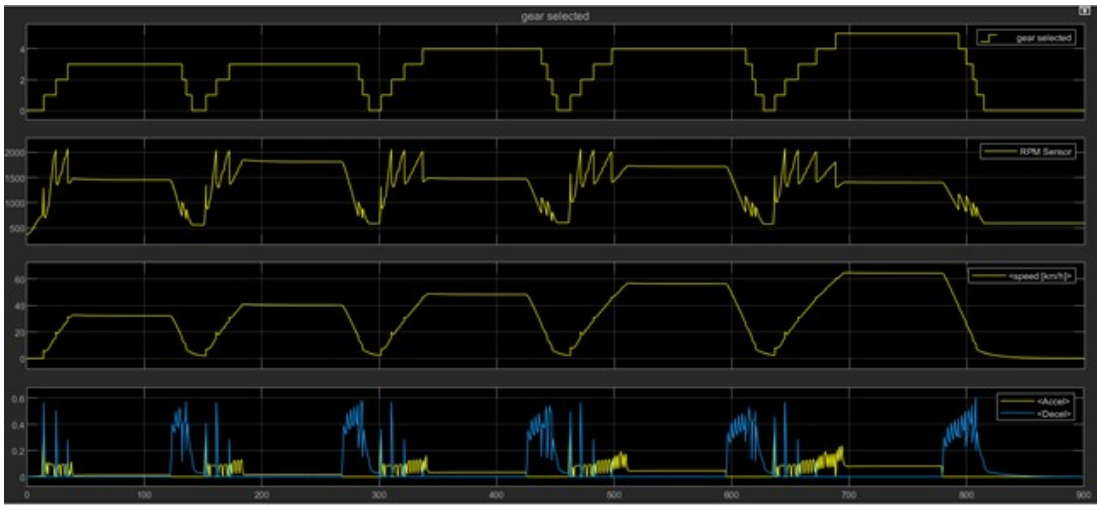

Figure 4.19: Results of Simulation 3

Now the driver selects also the 5th gear in the last section, and clearly the engine speed in that section diminishes.

4.2 Simulation

As done for BEV model, a final simulation is run just for reference. This simulation is run after a tuning procedure in order to make the model a little bit more performant in terms of fuel consumption and maximum vehicle speed, but is not consequent to an optimization process and doesn't want to be a comparison with simulations done with OLT. It is just done in order to assess how different components of the model

work together. As done for BEV, a table of parameters to set the model is shown on fig.

Parameter	Value	UM
Driving Cycle	WHDC	[-]
Mass	3500	[kg]
Wheels per axles	2	[-]
Cog-Front wheel dist	1.5	[m]
Cog-Rear wheel dist	1.5	[m]
Cog Height from ground	0.55	[m]
Frontal Area	4.5	[m²]
Cx	0.31	[-]
Rolling Resistance Coeff.	0.00896	[-]
Driveshaft Inertia	0.32	[kg×m²]
Rolling Radius	0.2032	[m]
Wheel Inertia	2	[kg×m²]
τ Final Drive	2	[-]
η Final Drive	0.98	[-]
Gearbox Inertia	0.15	[kg×m²]
Transmission Shaft Inertia	0.05	[kg×m²]
Engine Shaft Inertia	0.05	[kg×m²]
τ First Gear	6.02	[A⁄hr]
τ Second Gear	3.32	[-]
τ Third Gear	2.07	[-]
τ Fourth Gear	1.4	[-]
τ Fifth Gear	1	[-]
τ Sixth Gear	0.79	[-]
η First Gear	0.9675	[-]
η Second Gear	0.9675	[-]
η Third Gear	0.9685	[-]
η Fourth Gear	0.971	[-]
η Fifth Gear	0.99	[-]
η Sixth Gear	0.98	[-]
ICE	F4H	[-]
ICE inertia	1.44	[kg×m²]
ICE idle speed	750	[rpm]

Figure 4.20: Parameters for CV Simulation

ICE is the one given by OLT archives called "F4H" [**?**] and max power curve and fuel consumption map are generated by datidelveicolo.m. For what the driver parameters is concerned, the ones shown on section "Vehicle Body and Driver" are used. About ICE controllers, parameters shown on ICE section are used.

Once parameters are correctly updated, the simulation is run and results are shown in next figures.

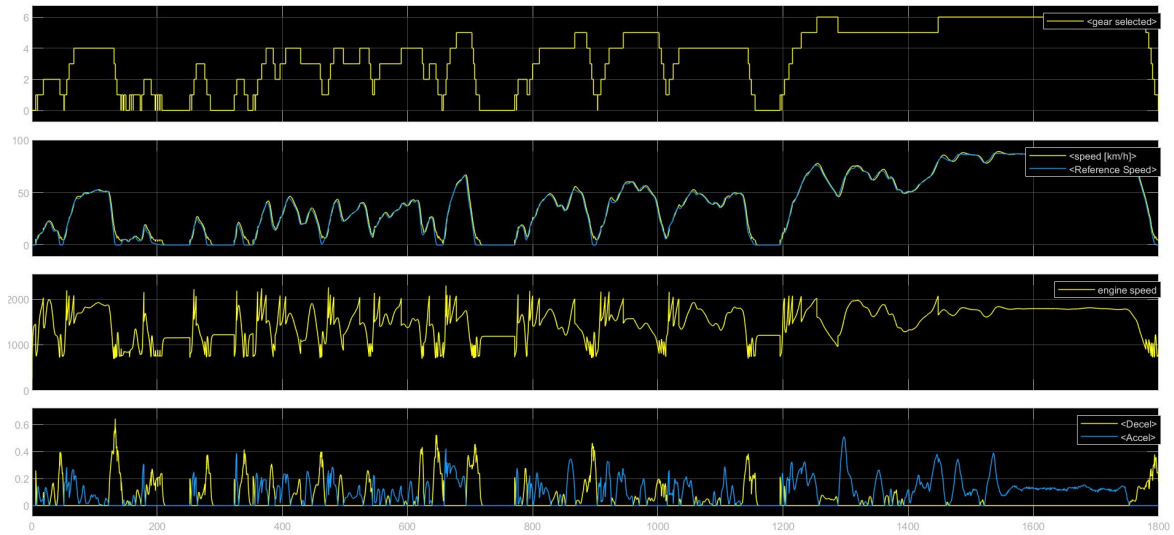

Figure 4.21: Main Simulation Results: From top to bottom: Gears, Vehicle Speed, ICE Speed, Pedal Profiles.

It is clearly visible that the driver correctly uses all gears, especially in the last part of the cycle in which the max speed is reached. The vehicle speed profile roughly follows the reference one, while the engine speed is always correctly bounded inside its intrinsic limits.

Also a Cumulative Fuel Consumption curve obtained integrating with Simulink operator the Instantaneous Fuel Consumption signal given by the ICE block. This curve is plot. in fig. 4.22.

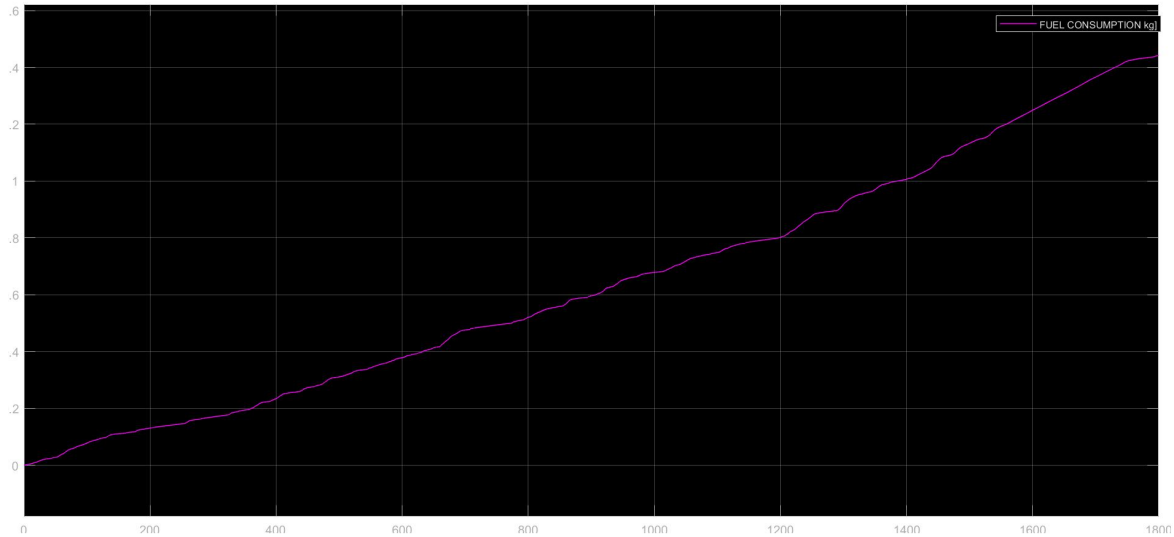

Figure 4.22: Main Simulation Results: Fuel Consumption Cumulative Curve [kg].

This trend is fundamental to compare engine performance simulation by simulation, especially its final value. Clearly an optimization procedure esulates the work of this thesis, but this curve is taken into account deeply when the comparison with GT-DRIVE model is performed in order to assess the performance of ICE and

precision of the model. It is somehow similar to the SOC curve for what the BEV model is concerned.

Chapter 5

Hybrid Electric Vehicle

This chapter is focused on designing a Hybrid Electric Vehicle model on Simulink / Simscape environment, basing on what has been done so far in Conventional Vehicle and BEV chapters, trying to use as much as possible same components and strategies shown in the previous chapters. When possible same blocks will be used, eventually integrated or modified. Biggest difficulties are expected in designing the control logic for the HEV, in particular for what BEMS and torque splitting is concerned. For this reason the first part of the chapter will be focused on just creating the model without a proper control strategies, but implementing something very simple just to check that all the electric and thermic components work together well. To import data from matlab, another matlab file is created, named as "datiHEV.m". From this file as usual is possible to import and modify parameters for customizing the HEV model.

5.1 Model Creation

To create the model, another Simulink blank model is opened and many subsystem representing each one a particular component, are inserted. First design choice is which hybrid electric architecture adopt to design our system. Since in OLT many architectures can be implemented, it is chosen to design a P2 architecture, which is considered the most used and flexible. P2 is a parallel hybrid architecture in which the linking between the electric power line and the thermal power lines is realized in between engine and transmission. It is possible to easily decouple the two power lines through a clutch coupling. Power flows from battery to electric motor via an inverter and through a torque coupling device it is linked to the ICE. In our case all the power is transferred towards the front axle, passing through a manual transmission and a final drive. The P2 scheme present in OLT archive and used as reference for the creation of the model is shown on fig. 5.1

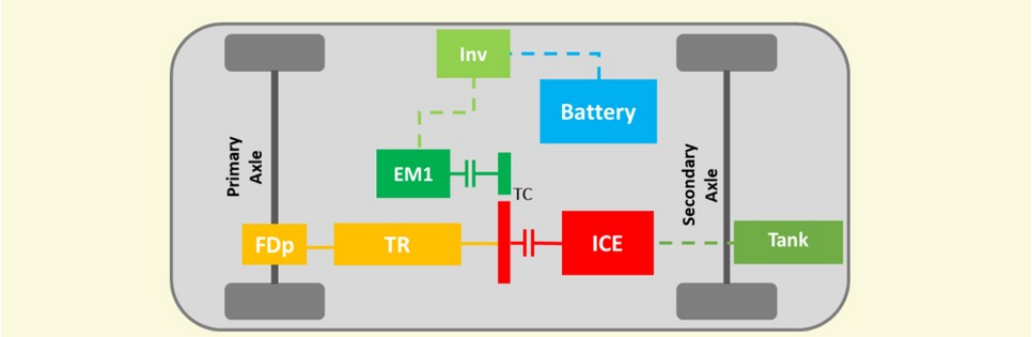

Figure 5.1: HEV Parallel P2 Architecture [**?**]

In the following subsections, each subsystem is briefly shown. During design of the model, many of these have been modelled being as much similar as possible with respect to previous models. By the way, some modifications have been necessary and some new components have been designed.

5.1.1 Vehicle Body and Brake Subsystem

Vehicle body subsystem is the same used in BEV and CV models

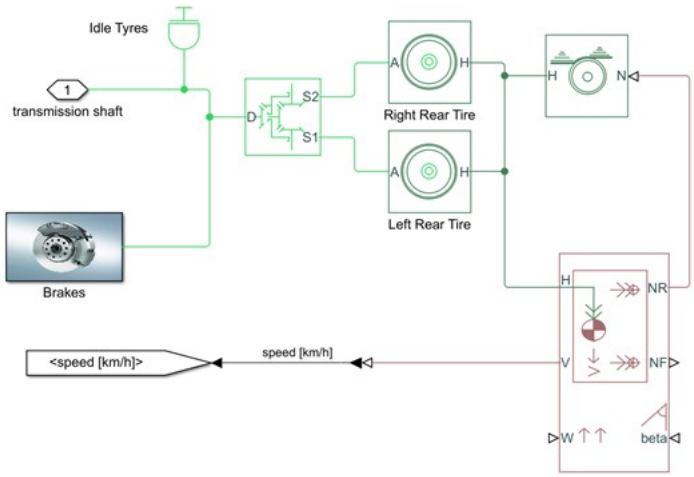

Figure 5.2: HEV Vehicle Body Subsystem

About brake subsystem, it is directly imported from the BEV model, because part of the braking torque will be given by brake, part by the electric motor, according to a braking strategy that will be later on implemented. This latter is a little bit modified with respect to the one used in BEV, as will be explained in the Electric Motor subsection.

5.1.2 Transmission

The transmission layout is very similar to the one derived by the conventional vehicle architecture:

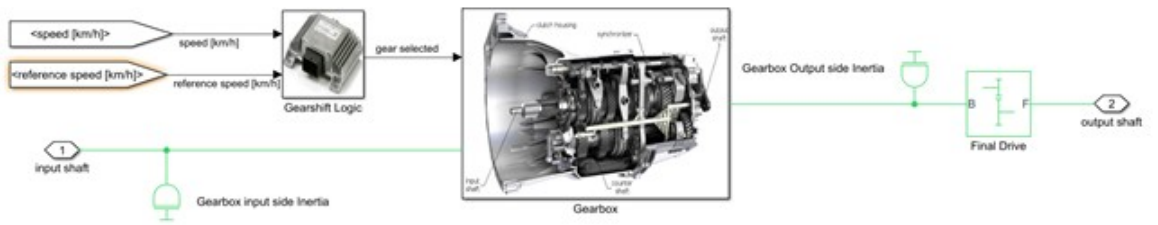

Figure 5.3: HEV Transmission

Final drive value is now called “fdp” on matlab and represents the “FDp” block in the reference scheme. Also gearbox is derived from CV model. What has changed for this simplified and initial model is the gearshift logic, which is still produced by a chart stateflow, but now, since we will have more than one torque provider, is based on vehicle speed. In fact if we based our gearshift logic on ICE speed, we would constraint also the electric motor speed, resulting in a non correct gear selection. The stateflow chart is the following one:

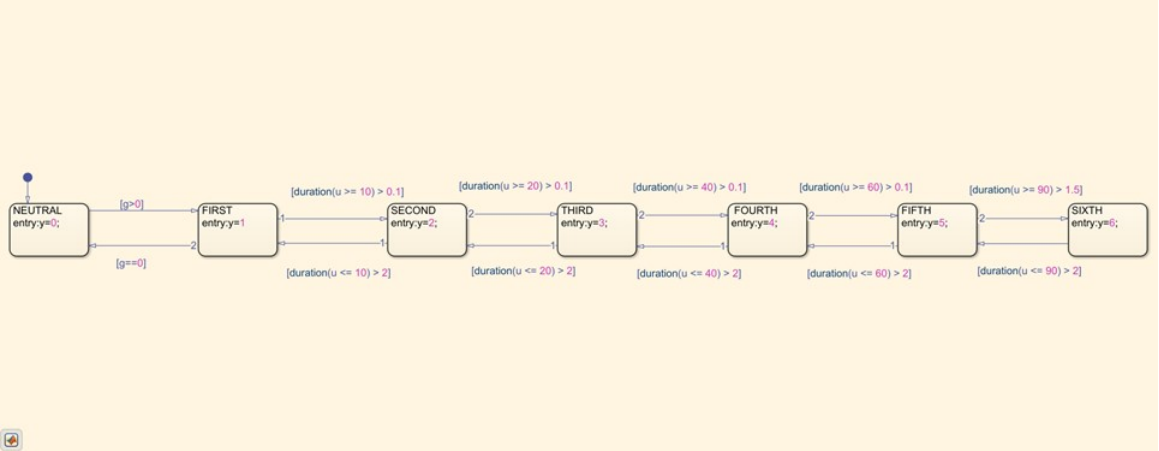

Figure 5.4: HEV Gearshift Logic

To pass from neutral to first it is mandatory just that the reference speed is greater than 0, while gearshift is led only by vehicle velocity threshold speeds.

5.1.3 Electric Powerline

Input shaft of the gearbox is now not directly associated with electric motor neither ICE. It must be connected with both of them. Looking at the P2 scheme present in OLT, it is mandatory to realize, for what the electric powerline is concerned, a torque coupling device, which is just a simple gear block, in charge to reduce the speed of the shaft linked to the electric motor. Value of gear ratio is specified in matlab. Also a clutch is necessary to make possible to decouple/couple the transmission and the

electric motor. This clutch will be commanded by an actuator very similar to the ones inside the gearbox. The actuator will be triggered by a command coming from a controller, which by now always sends a 1 signal. It means that for the moment the electric motor will always take in charge to move the vehicle. The B port of the clutch is instead clearly associated with the electric motor, taken from the BEV model. The model is therefore by now so composed:

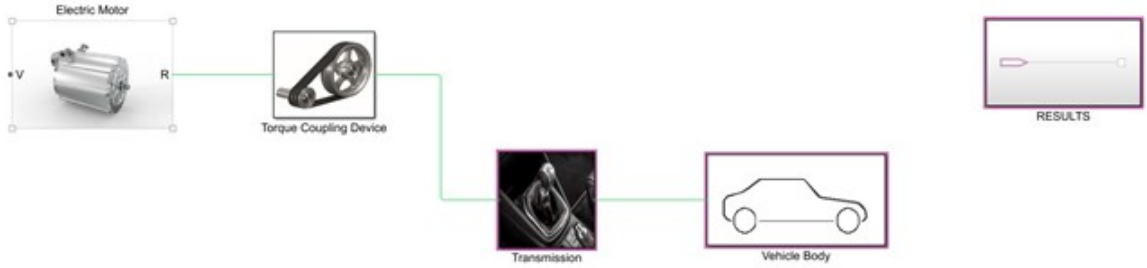

Figure 5.5: HEV Electric Powerline

Inside the torque coupling device:

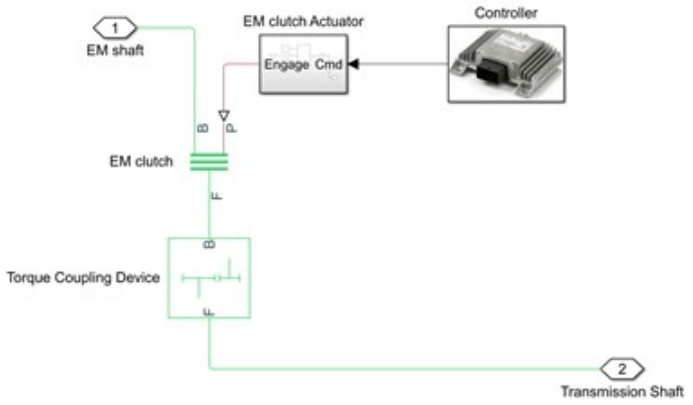

Figure 5.6: HEV Torque Coupling Device

Inside the electric motor block, everything is taken from the BEV model. A difference is that for the moment the accelerator pedal is divided by 3, since the idea is to split 33/66 the torque demand by the driver, for the initial stages of the non controlled model .
In addition to this, a little modification must be applied to the braking system with respect to the BEV model. In fact in the latter the electric motor brakes only up to 100 N*m when the brake pedal is pressed. The rest is given by brakes. What now we want to implement in our HEV model is instead that all the adsorbable torque is given to the electric motor, and the rest is covered by brakes. But the motor cannot always adsorb 300 N*m, because the limit curve displays that in some situations in which the speed exceeds some values, the maximum torque is reduced. So we need

to implement a threshold of torque which is variable and depends on electric motor speed itself. For doing that we use the block "dynamic saturation", in which the upper limit is the maximum torque got by the torque-speed envelope of the engine, while the lower limit is 0. Now the situation is the one depicted below:

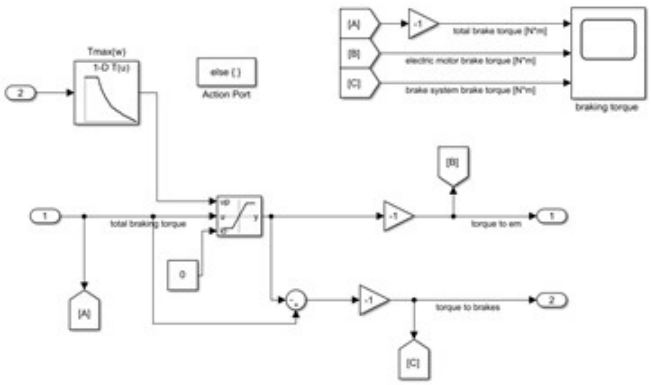

Figure 5.7: HEV Braking Control Logic

The saturation limit is not now anymore set to 100 N*m constant value, but continuously changes basing on the rpm speed (inport 2) and the look-up table. In this way the car in braking phase will always brake with electric motor up to its limit, before using brakes.

5.1.4 Battery

The battery is the one taken from the BEV model:

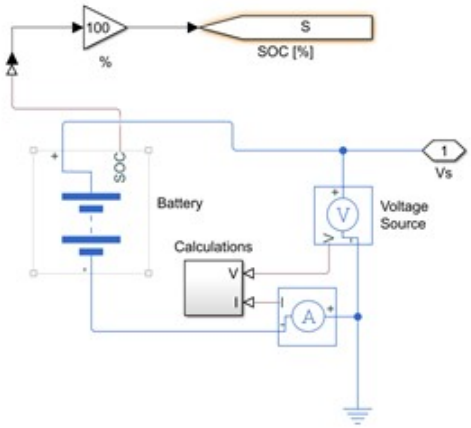

Figure 5.8: HEV Battery

5.1.5 ICE powerline

Looking at the reference image, we need another Clutch to link the ICE to the driveline. The same approach done with the torque coupling device is used, with the difference that no gear ratio is necessary in this case. Also in this case a controller is

inserted, which for the moment always engages the clutch. Everything is put inside a block called "ICE clutch" which is so composed:

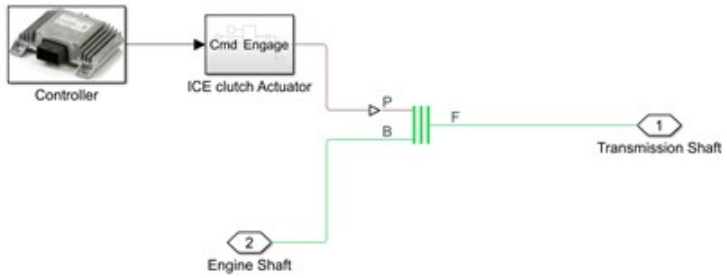

Figure 5.9: HEV ICE controller

For what the engine is concerned, it is simply taken from the CV model. Also in this case the throttle command is halved, since we have to 50/50 split the torque. The model is now so composed:

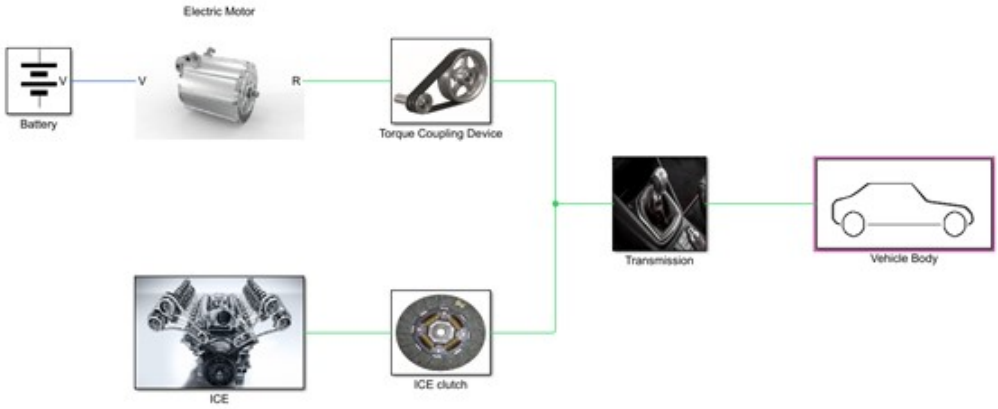

Figure 5.10: HEV Model

As we can see it is a P2 architecture, because the two power lines gather before the transmission.

5.1.6 Driver

What is now missing is the driver, which is taken from the previous model BEV or CV. Final model so is:

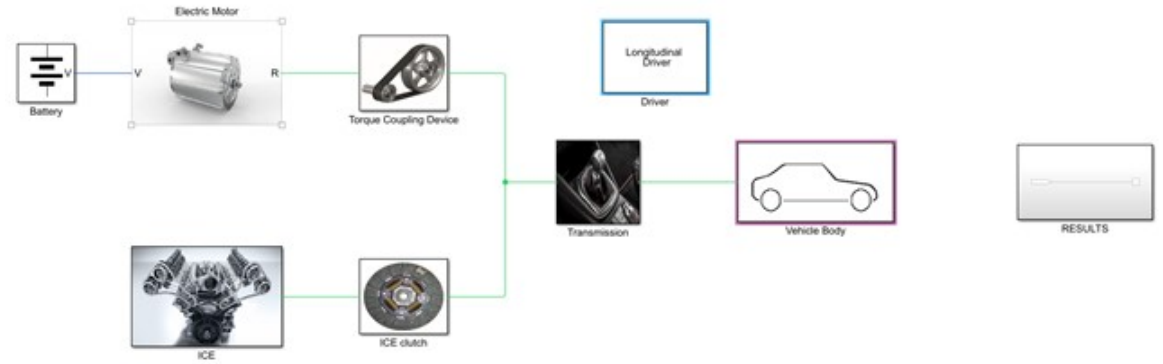

Figure 5.11: HEV Model

5.2 Tuning Simulation

We want to run a basic simulation of the model in order to just verify that the non-controlled model is able to accomplish a simple cycle. Since as already said the model is not controlled, we for the moment impose that 1/3 of the throttle signal will be given to the ICE, 2/3 to the Electric Motor. The cycle is imported by an excel file created on purpose, which is shown below:

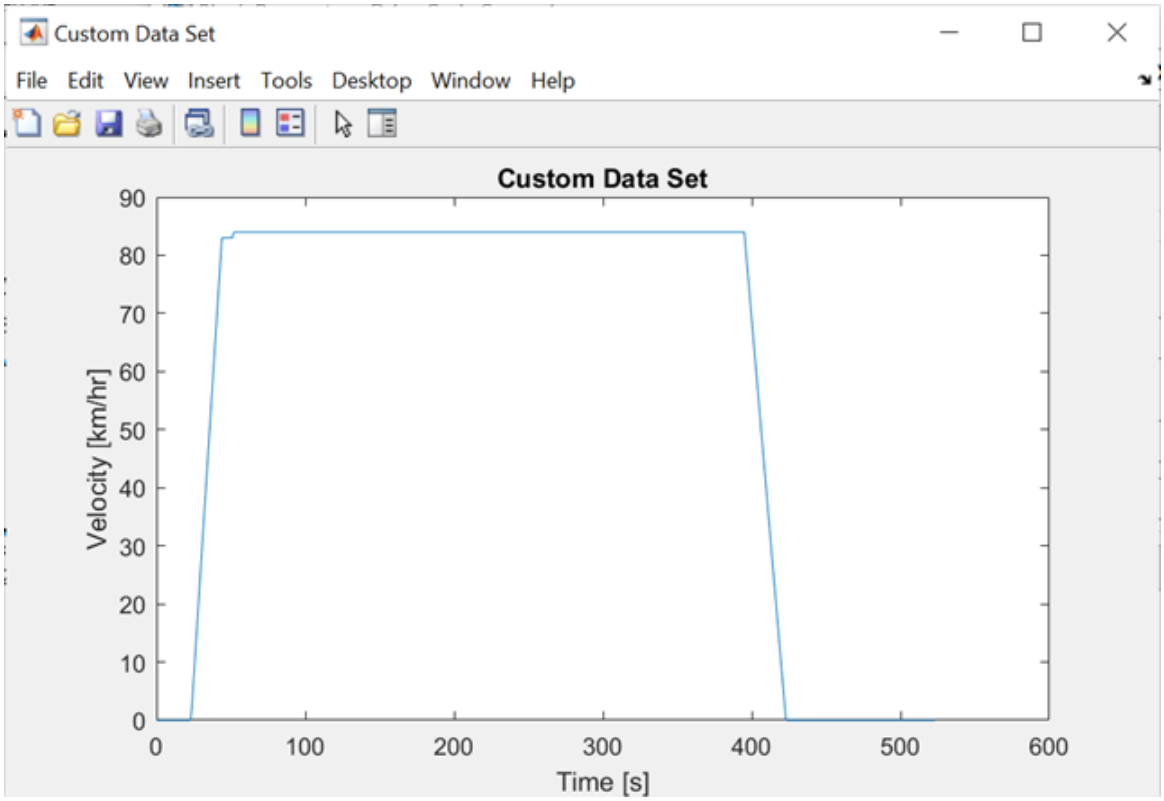

Figure 5.12: HEV Basic Cycle

Before running the simulation, some adjustments must be taken. In particular:

- Gearshift logic: Another little modification must be applied, related to conditions in order to pass from first to neutral gear: we have to impose that this downshift verifies if and only if both reference speed and braking signal are both to 0. This is due to the fact that if we have a little delay in the braking phase, which basically means that the car should be at rest but the driver is still pressing the braking pedal because the car actually is not stopped yet, in the old configuration the neutral gear would be selected while the car is still braking, causing that the electric motor would receive a braking signal without feeling the inertia of the vehicle. Experimentally this means that the electric motor inverts its speed and rapidly goes beyond its limit, causing the end of the simulation. In addition to this, the vehicle would not brake anymore. To fix this problem we have to, as already said, select the neutral gear only if the vehicle should be at rest, and actually is. This makes totally useless the electric motor clutch, which should be always engaged (also in thermic traction only), when the driver is going to brake. The only situation in which we can disengage the electric motor clutch is when we are in thermal mode and we are not using brakes, which by the way is not a situation taken into account in this initial not controlled HEV model.
- Thermic actuation: it is not possible to run the simulation always keeping the ICE clutch engaged. Clearly the ICE is not ductile as the electric motor, but it has to run always at a minimum speed, also when the vehicle is not moving. This problem is basically fixed in a CV using the neutral gear, so using the gearbox clutch. This is not possible in this HEV model, because even if we select the neutral gear when the vehicle is at rest (which basically happens), the engine would be however linked to the electric motor, dragging it and recharging the battery, consuming fuel, also when it is not required or asked. The only way to avoid this phenomena is to design an easy control logic of the ICE clutch via Stateflow. The logic is depicted in 6.6: In this way when we are at rest, the ICE clutch will detach the ICE from the electric powerline, leaving the car in a sort of “electric mode”. The clutch will be again engaged when car needs to be moved, entering again in a “hybrid mode”. The signal coming from the stateflow chart is sent to the ICE actuator, in order to control the ICE clutch.

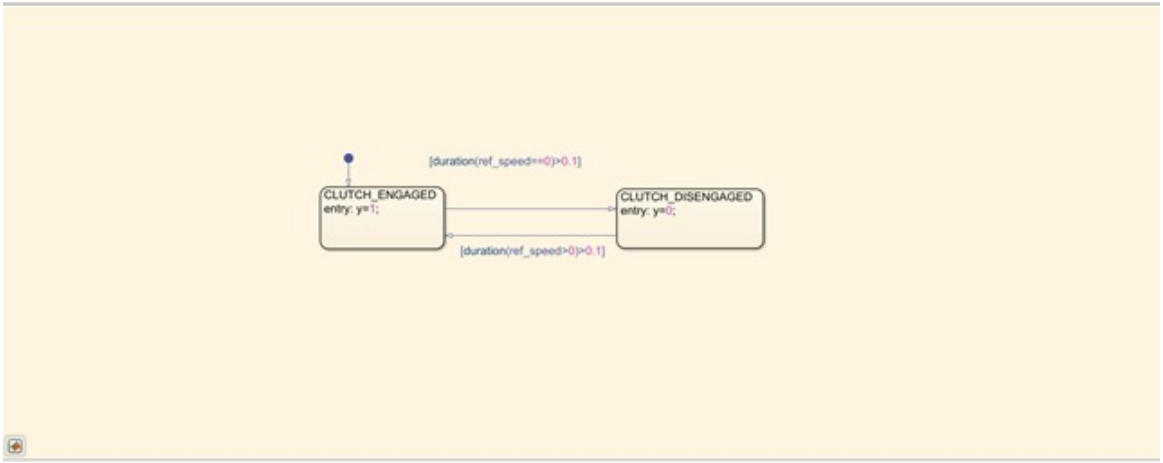

Figure 5.13: HEV ICE Clutch Controller

- Idle Speed controller: the Idle speed controller needs to be strongly modified. Running the simulation with old data, at the end of the braking maneuver, when the ICE clutch is disengaged, the ICE idle speed controller would strongly accelerate the engine, increasing its speed beyond limits, killing the simulation. An accurate tuning is done changing threshold parameters of the controller, which are selected as depicted below:

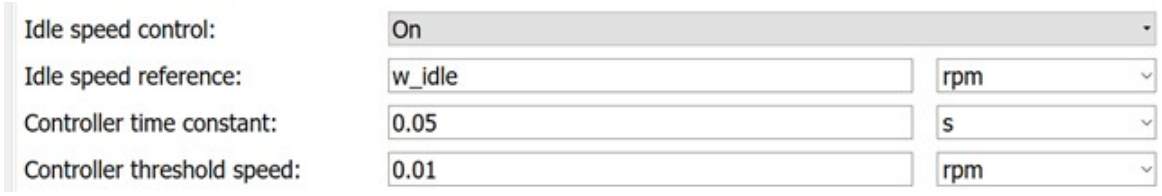

Figure 5.14: HEV ICE controller parameter

- Also for what the PI parameter of the driver, they need to be accurately tuned to avoid driver presses inappropriately pedals:

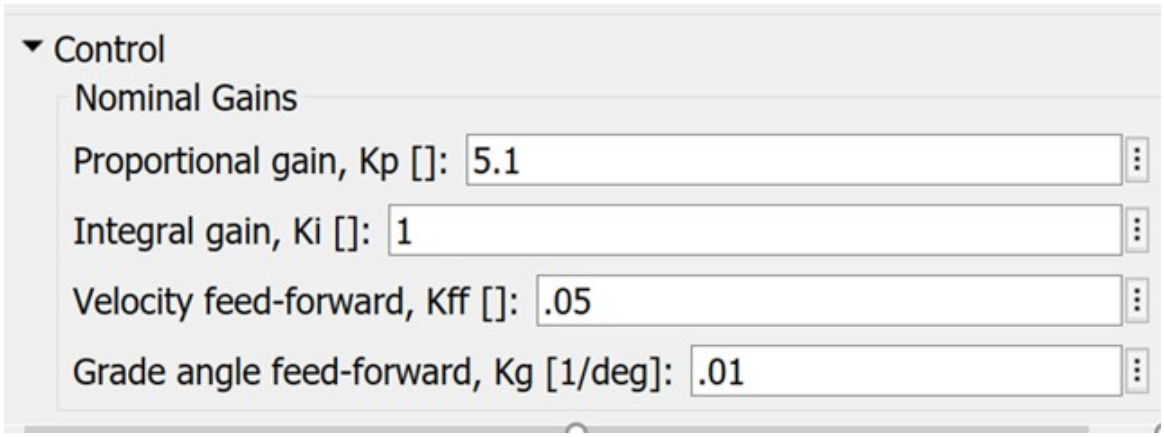

Figure 5.15: HEV Driver Parameter

- Initial SOC value is imposed to 0.7

Now the simulation could be run and 5 scopes are created in the section "Results":

1. Vehicle Parameters: speed and reference speed comparison, gear selected, throttle and brake comparisons.
2. EM Parameters: torque, speed, power, clutch engagement
3. ICE Parameters: speed, power, clutch engagement, fuel consumption
4. Battery Parameters: SOC, current and voltage, Battery Power, Joule effect dissipated Power.
5. Motors Parameters: ICE speed, EM speed, tractions, power comparisons.

The results are shown below:

Vehicle Parameters

Figure 5.16: HEV Vehicle Parameters

We can note that the driver is slightly able to follow the cycle, even if it overcomes the reference speed in the first part of the cycle. It properly selects all the gears up to the 6th, and at the end of the cycle it properly selects also the neutral. For what the pedals are concerned, driver properly accelerates at the initial stages, brakes at the end.

EM Parameters

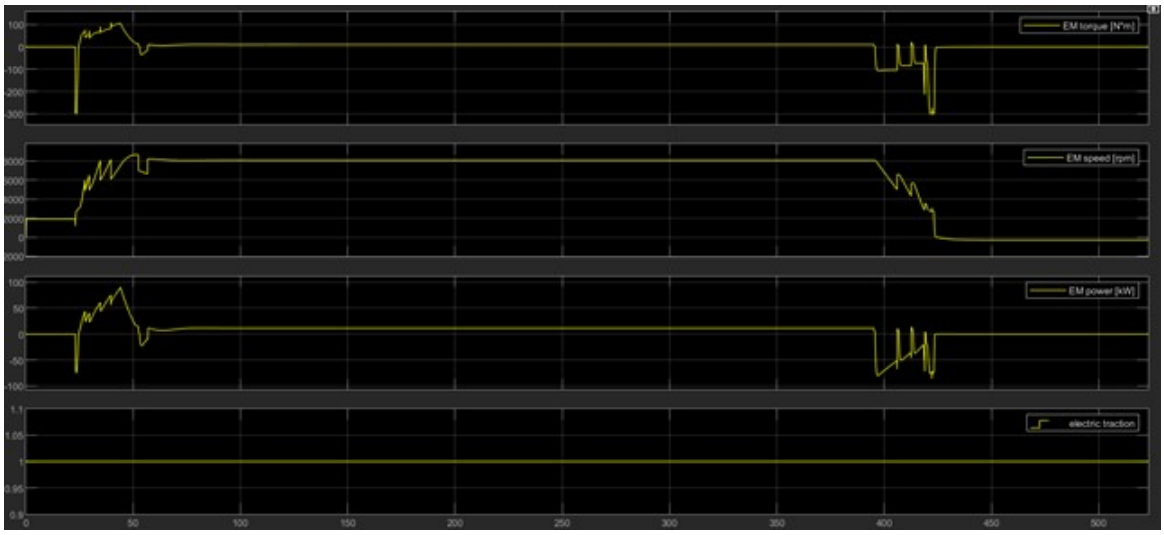

Figure 5.17: HEV EM Parameters

For what the EM is concerned, we can see how the torque demand is always inside the boundary limits, positive in the acceleration phase and coast phase, negative in the deceleration phase. Also speed has a correct path, approaching 0 at the end of the cycle. Same considerations hold for power, while traction is always electric (1 means clutch engaged), for reasons already explained.

ICE Parameters

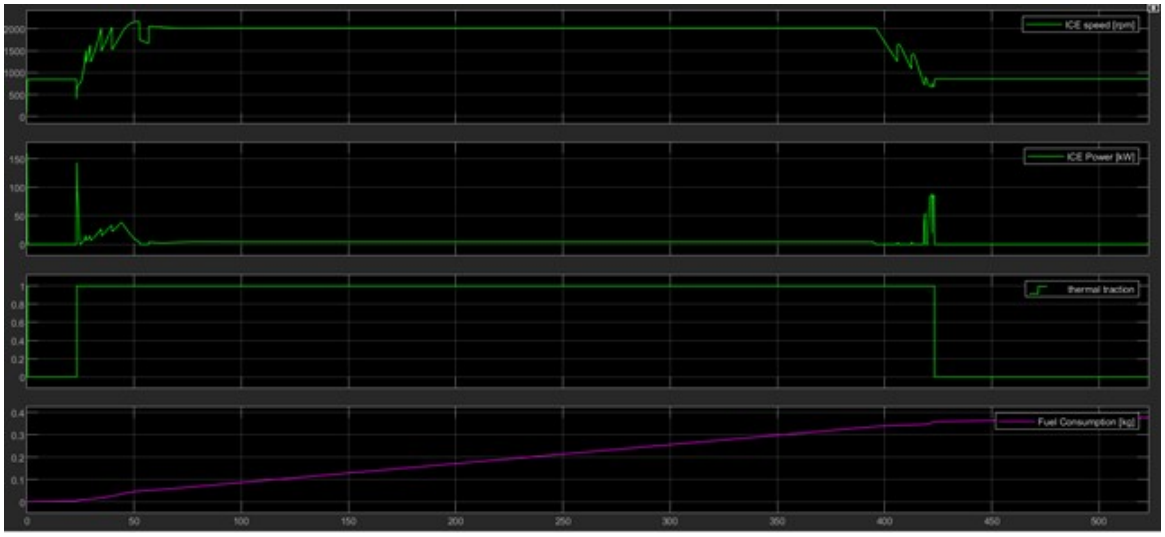

Figure 5.18: HEV ICE Parameters

For what ICE is concerned, we can see how the speed is inside the boundary limits, and never goes below the idle threshold (expect for a negligible spike due to clutch ideally modelled). Power is requiring for ICE during the acceleration stage and during the initial stage of deceleration, in which because of a little lack in disengaging the ICE clutch exerted by the controller, for a short interval of time it is connected to the electric motor and recharges the battery. The clutch is immediately disengaged after that and the power is correctly brought to 0. Same considerations hold for Fuel Consumption.

Battery Parameters

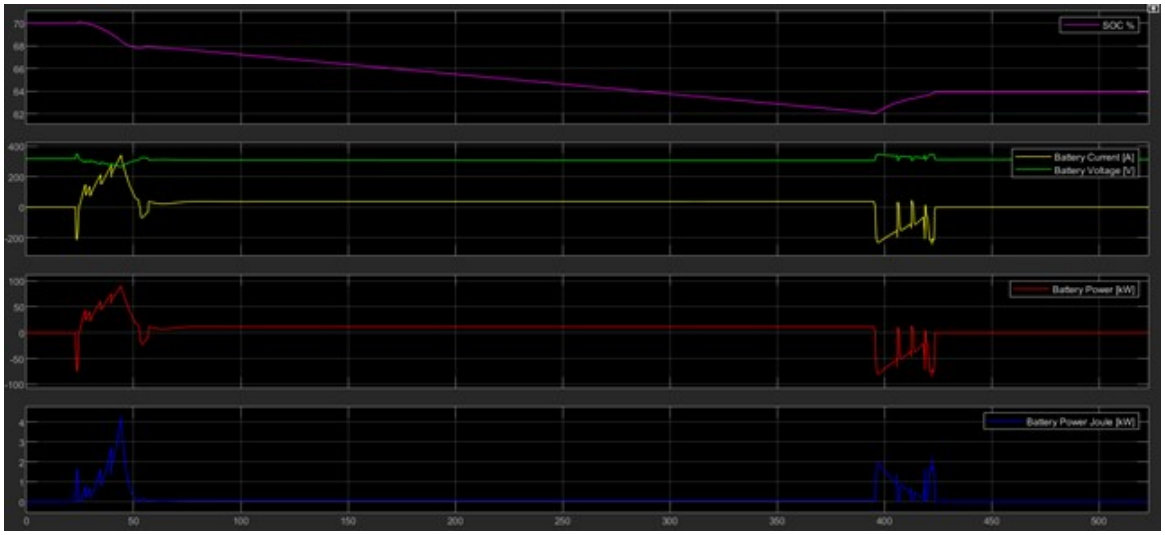

Figure 5.19: HEV Battery Parameters

For what battery is concerned, we can see how the battery correctly discharges during the acceleration and coast phase, and recharges in the braking phase, also

due to that little length of time in which the engine drags the electric motor. Voltage is correctly always around 300 V, while current and Power graphs are correctly simil shape. Also losses for Joule effect can be appreciated, which are however very small.

Motors Parameters

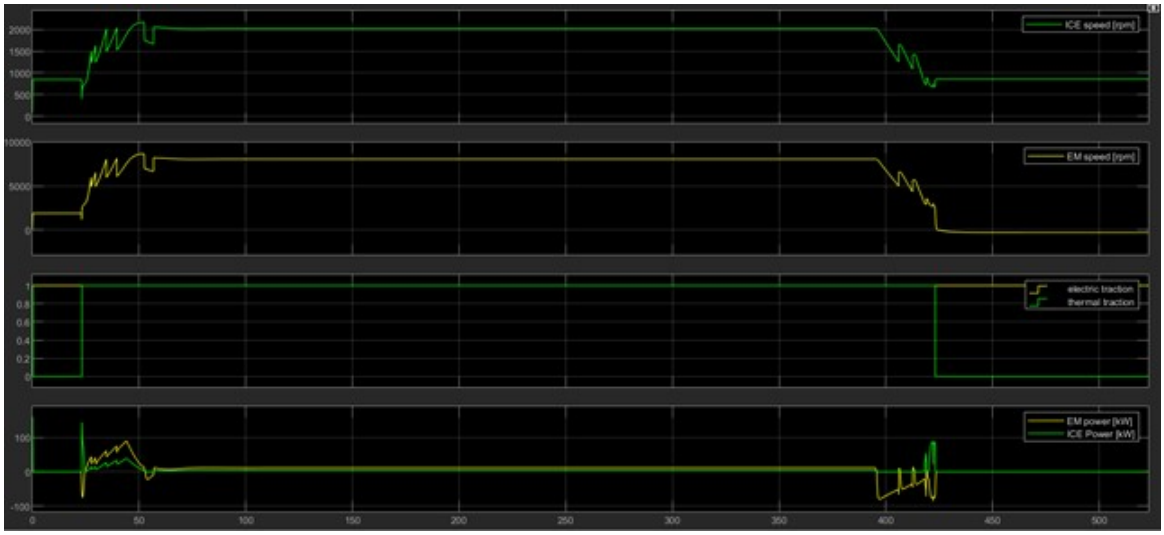

Figure 5.20: HEV Motor Parameters

A last graph can be interesting in analysing how during the stages of the cycle the two propulsion lines interact. It is possible to note at the end of the deceleration phase, prior the ICE clutch is disengaged, the mirror shaped power curves of the two ICE and EM, which means the first is dragging the second one.

5.3 ECMS Design

This part of the chapter is focused on the design of the Equivalent Consumption Minimization Strategy for HEV and has been realized in collaboration with Riccardo Russo. Indeed the model designed so far is a non controlled one, in which the splitting in torque between the electric and thermal powerline is fixed to 33/66. This choice has been taken in order to realize a simple hybrid traction and verify that the system worked, but doesn't exploit the benefits of this architecture. To properly use the hybrid vehicle it is necessary to design a controller that instant by instant selects the best solution for what the splitting of the torque is concerned, when the vehicle is in acceleration phase. To do this, an ECMS controller is designed in Simulink / Matlab environment. [14] This controller is inserted in the non-controlled HEV model. A "Matlab Function" block is inserted in the Controller subsystem which was created to allocate the ICE clutch controller with StateFlow.
Fig.5.21 shows the matlab function that represents the controller, with its numerous input and one output.

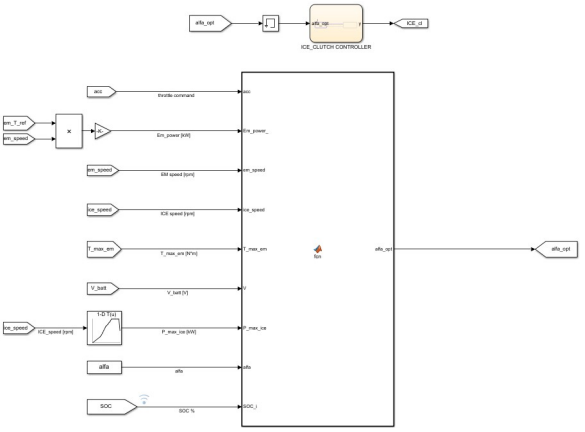

Figure 5.21: HEV ECMS Controller

The output of the block is the optimum value of alfa, which is the splitting in torque between the electric motor and the engine and is so defined:

$$\alpha_o = \frac{T_e}{T_e + T_i}$$

with T_e and T_i the torque chosen by the controller to be sent to the electric motor and the engine respectively.
Inputs of the block are many, starting from the top:

- Throttle Command given by the PI controller.
- Power in input to the electric motor, computed as product of the instantaneous rotor angular speed and the torque coming out of the electric motor controller
- Electric motor rotor speed
- Engine crankshaft speed
- Maximum torque allowable by the electric motor at that particular speed, computed by means of a look-up table with electric motor maximum torque map implemented inside.
- Voltage of the battery, coming from the battery subsystem
- Maximum power allowable by the engine, computed by means of a look-up table with engine maximum power map implemented inside.
- An α_i vector representing all the possible splitting in torque.
- State of Charge of the battery.
- Efficiency of the battery.

Basic idea is to compute instant by instant a vector of "hybrid fuel consumption" that the HEV would experience if different possible values of torque splitting are selected. These "hybrid fuel consumption" values are computed as sum of the instantaneous fuel consumption of the engine, and an equivalent fuel consumption

of the electric motor. Instant by instant the algorithm selects the smaller value of these "hybrid fuel consumption", returning the correspondent value of α_o as output. First of all it computes:

1. The Power required by the HEV in traction as a part (depending on the throttle) of the maximum power allowable by engine and electric motor together
2. The Power required by the HEV in braking maneuvers as the Power of the electric motor coming out of the braking splitter controller of the electric motor.

Then it computes six possible values (depending on α_i) of torque demand to the electric motor, and with the efficiency map, six possible values of efficiency η_e . From this efficiency, the algorithm is able to compute six values of sort of equivalent electric fuel consumption q_e, dependant on a s_d factor, called "discharge factor", which depends on the driving cycle the HEV is going to make. This s_d represents a sort of "penalty" that the electric traction represents with respect to the thermal one.
Same procedure is done with the internal combustion engine, resulting in the real six q_i values of the instantaneous fuel consumption.
Then six values of the "Hybrid Fuel Consumption" J function are computed simply as a sum of the two fuel consumptions previously computed:

$$J = q_e + q_i$$

At the end the algorithm chooses the minimum value of the J function and returns its correspondent α_i value, which is the α_o.
When the vehicle is braking, the algorithm is forced to return an $\alpha_o = 1$ and allowing the regenerative braking so as to designed by the braking splitter.
In addition to this, the algorithm is designed so that to guarantee that the SOC of the battery doesn't fall down a minimum value $SOC_m = 0.6$. Instant by instant the SOC(α_i) value that the battery would have with each α_i is computed, and if $SOC(\alpha_o) < SOC_m$, an α_o=0 is returned, which basically means the vehicle is giving up on using the electric powerline because the battery is discharging too much. With these constraints, we are pretty sure that:

$$SOC > SOC_m$$

Once the value of α_o is chosen according to what has been previously said, signals of acc_i and acc_e throttle accelerator commands for respectively engine and electric motor are computed directly in Simulink environment in this way:

$$acc_i = acc * (1 - \alpha_o)$$

and

$$acc_e = acc * \alpha_o$$

in which acc represents the throttle signal coming out from the PI driver.

Before to show the final HEV controlled model, a small modification needs to be applied to the ICE clutch controller. As already explained, this controller uses a

Stateflow logic in order to engage or disengage the clutch. In non-controlled HEV model, this logic was based on the reference speed of the cycle. This was acceptable because, since the splitting was fixed to a permanent hybrid mode, the ICE was always required to give power and so the fact that the clutch was disengaged only when the vehicle was at rest was pretty comfortable.
By the way in the controlled HEV model, we hopefully expect that more than once during the cycle the HEV is supposed to go in pure electric mode, as suggested by the ECMS. In this case, with the old ICE clutch controller with Stateflow logic, the clutch would be kept engaged, resulting in a useless dragging of the engine. For this reason, the stateflow logic is updated, sending engaging signal only when $\alpha_o < 1$. The updated clutch controller is shown in the next figure:

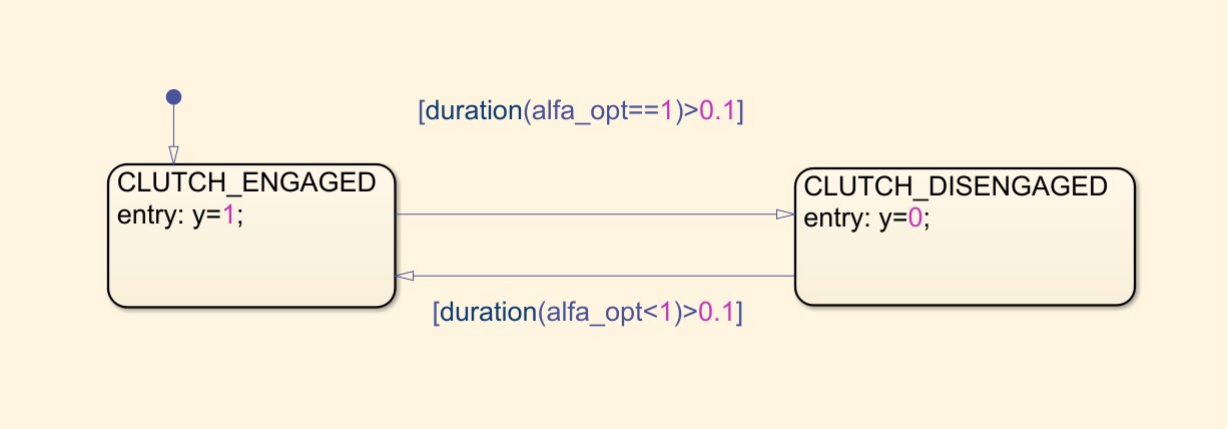

Figure 5.22: HEV Updated ICE Clutch Controller

The final HEV model, which now is a controlled version, is shown in fig. 5.23.

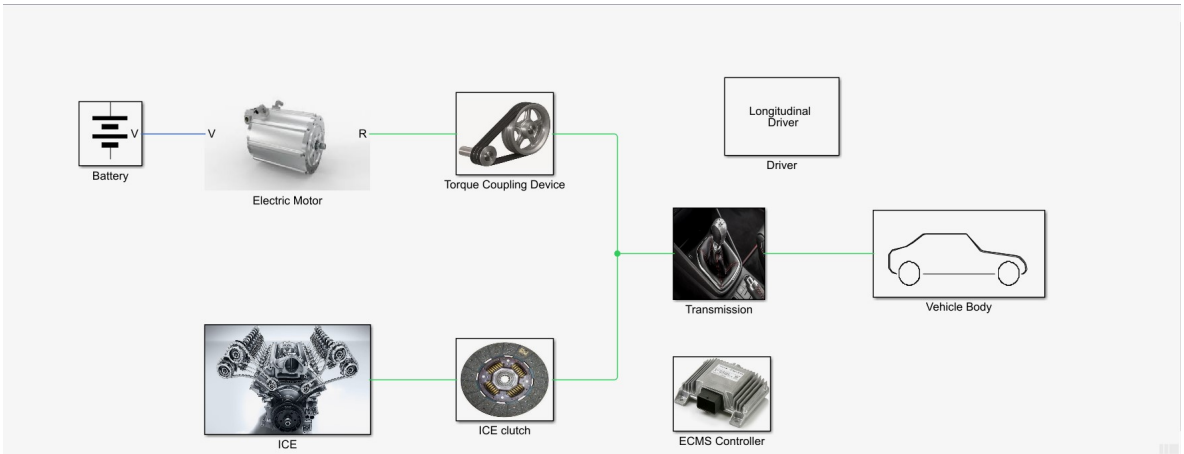

Figure 5.23: HEV Final Controlled version with ECMS Controller

5.4 DP-ECMS Comparison

One of the most important aim of this thesis work is the comparison between theHEV Simulink model designed so far with an ECMS and the DP simulation coming from the Optimum Layout Tool (OLT) described in the first chapter. A simulation over

WHDC is run with valuable and plausible vehicle data, with both Simulink and the OLT.
In Simulink a tuning procedure to the s_d parameter is performed, so to find the right value in order to have at the end of the cycle the same value of initial SOC_i. Once this tuning operation is performed, the two simulation are comparable. What is expected to get as a results is that the DP simulation is the one which leads to less fuel consumed at the end of the cycle, since the strategy used inside the OLT is surely more optimal. fig. 5.27 shows the comparison between the two instantaneous value of SOC during the two simulations.

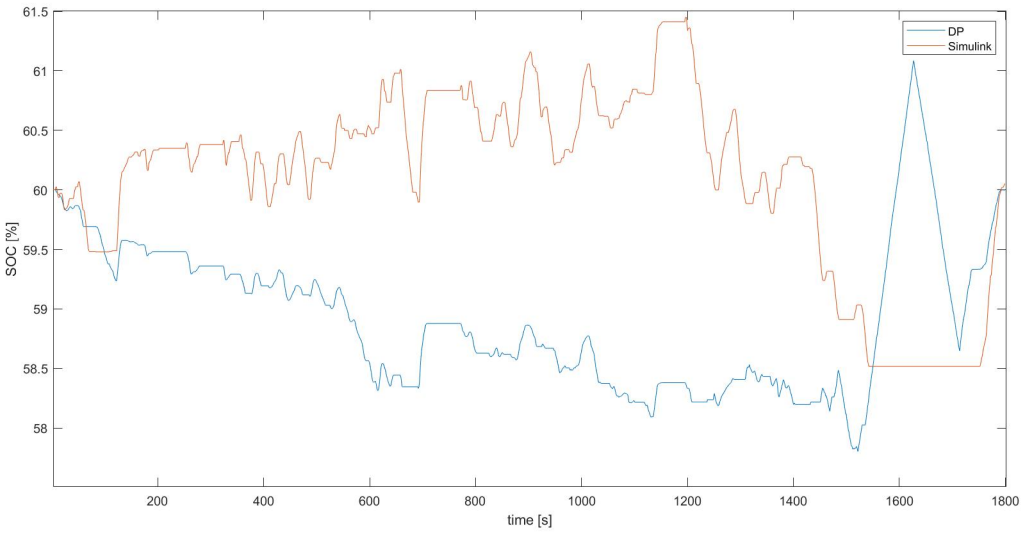

Figure 5.24: DP - ECMS Comparison SOC

we can see how the OLT discharges more the battery during the cycle and recharges it in the last cruise phase, while the ECMS strategy leads to use more the "hybrid mode" during the cycle, so to not recharge the battery with the engine. It is worth to note how both SOC final value are coincident, so the battery charge sustaining is guaranteed.
A very important parameter to assess the validity and performance of the two strategies is fuel consumed by the vehicle to run the driving cycle.

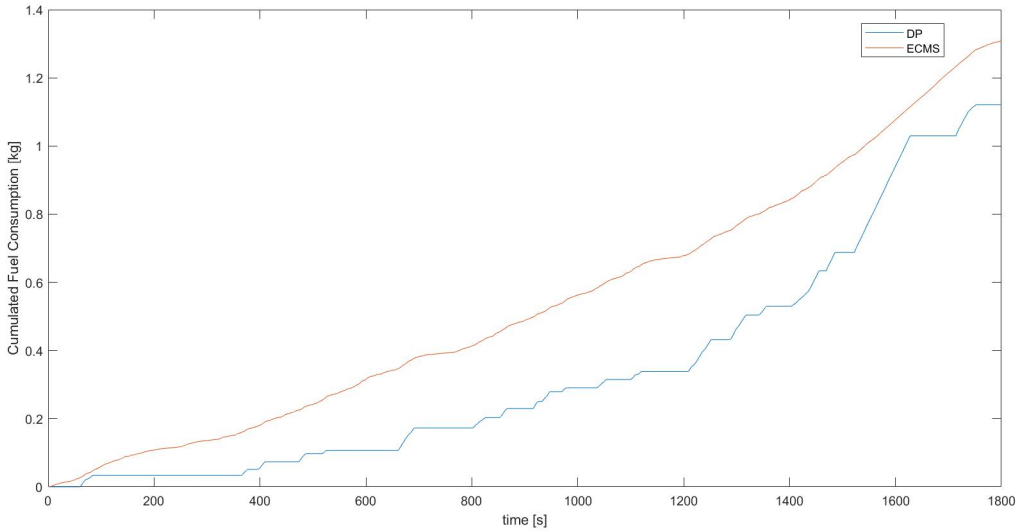

Figure 5.25: DP - ECMS comparison Cumulative Fuel Consumption

fig.5.25 shows the two cumulated trend of fuel consumption. It is possible to notice how the DP simulation globally optimizes fuel consumption, even if in the final phase of the cycle it uses the ICE to recharge the battery.

The table below shows the two final value of cumulated fuel consumed at the end of the cycle, in [kg].

Strategy	FC [kg]
DP	1.121
ECMS	1.309

Figure 5.26: DP - ECMS comparison Fuel Consumed

As expected, the DP simulation better optimizes the fuel consumption, even if results obtained with ECMS can be considered as very good. By the way the two values are very close one to each other, which confirms the validity of the two simulations.
Just for reference, also a last plot of α_o got by the ECMS during the simulation is shown, to appreciate the "hybrid mode" chosen during the acceleration phases of the cycle.

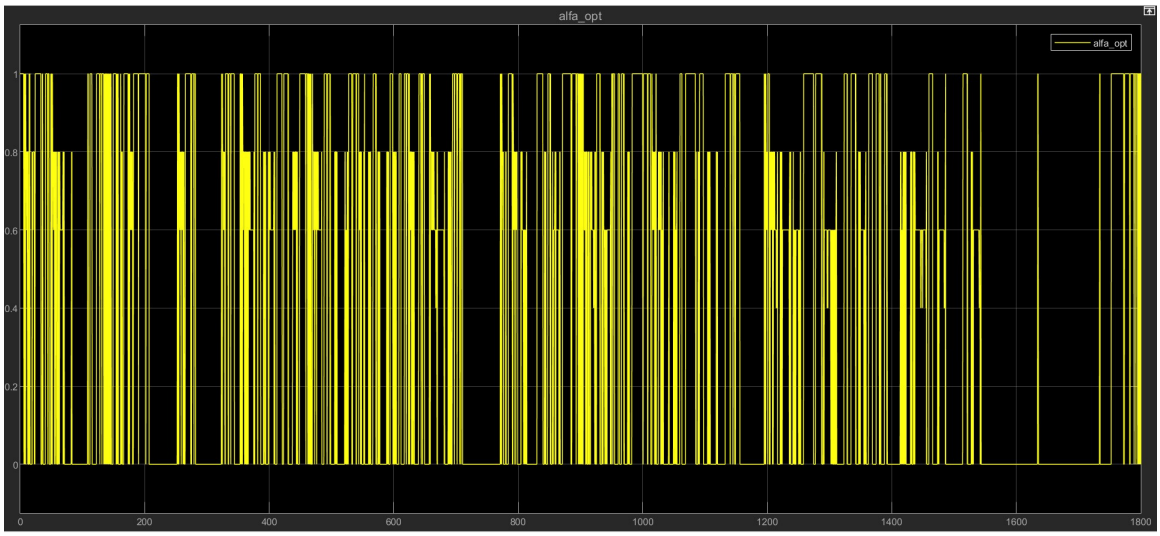

Figure 5.27: ECMS Torque Splitting

We can appreciate how many times during the cycle the ECMS chooses autonomously to give as output a hybrid torque splitting.

Chapter 6

Simulink GT-Power Comparison

Aim of this chapter is comparing models created on Simulink / Simscape environment with those created on GT-Power. This chapter has been written in collaboration with Riccardo Russo.
This work is intended to follow a first part developed in OLT, so, where possible, all data and settings were implemented in accordance to OLT data provided so to allow a comparison between the two models. The vehicles models compared will be of two different types, a battery electric vehicle (BEV) and a conventional diesel (CV) vehicle. This report will briefly analyse each component, showing analogies and differences between the two models, and finally show the results after a simulation on the same driving cycle for both the BEV and the conventional vehicles. Since all the parts' data have been taken from OLT provided data, where the electric propulsion system was intended for a HEV and not a BEV, in the case of BEV the electric motor would not be capable to propel the heavy duty vehicle considered in OLT, so a typical commercial vehicle has been considered in this case, since the final objective of this work is not the comparison with OLT results but among the two BEV models. The discussion will start with the BEV comparison followed by the conventional vehicles' one.

6.1 BEV

In each subsection, components will be analysed and compared.

6.1.1 Battery

Battery has been modelled in both cases as a map-based battery, characterized by open circuit voltage (Voc) and internal resistance (Req) in function of the battery state of charge (SOC), all provided from OLT data.

-	Simulink	GT-Drive
Nominal Voltage [V]	300	300
Overall number of cells	1680	1680
Number of cells in series	84	84
Number of cells in parallel	20	20
Battery capacity [Ah]	58	58
Battery Temperature [K]	-	300

Figure 6.1: Battery Parameters for Simulink and GT-Drive

For the sake of simplicity, only some values of reference vectors for look up tables for Vocv(SOC) and Rtot(SOC) are reported.

SOC	0	0.2	0.4	0.6	0.8	1
Req (Ohm)	0.241	0.161	0.149	0.149	0.155	0.152
Voc (V)	279.0	293.1	301.4	309.3	322.8	335.2

Figure 6.2: Battery Parameters for Simulink and GT-Drive

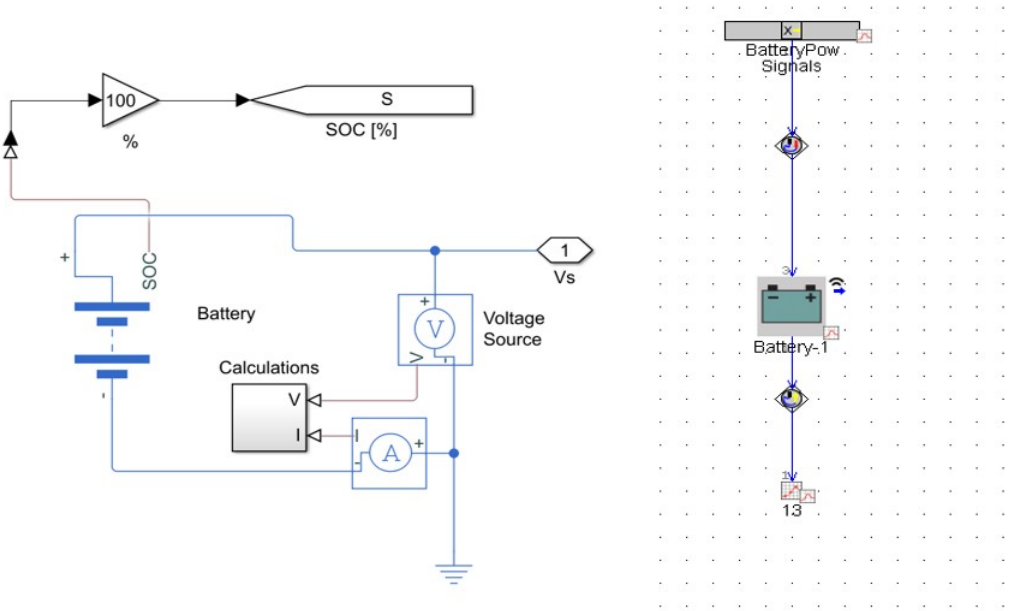

Figure 6.3: Battery Comparison models in Simulink (right) and GT-Drive (left)

- Both models set values of Vocv and Rtot dependant only on SOC and not on temperature, which has been set constant in GT-Drive model and not specified in Simscape.
- In both models Req map is the same in charge and discharge conditions.

- None of the two models take into account the self-discharge of the battery.
- In both models the interpolation method for Vocv (SOC) and Rtot (SOC) is linear.
- In both models, dynamic behaviour in charge and discharge is not modelled, meaning that the battery immediately responds to load variation.
- For both models no fade is modelled. It means that the battery doesn't deteriorate over cycles.

Regarding the computation of the Total Capacity of the Battery (A/h), numbers of cells in series and parallel and single cell capacity data have been used.

6.1.2 DC/DC Converter

In OLT, DC/DC converter was considered by just adding an efficiency of 0.95 to the battery, but no alterations in voltage are applied.

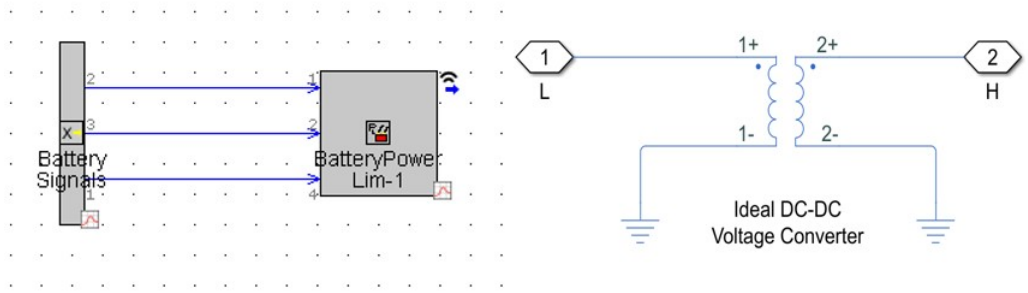

Figure 6.4: DC-DC Comparison models in Simulink (left) and GT-Drive (right)

In Simulink model of DC/DC converter is not possible to properly insert an efficiency, which, instead, can be done in GT-Drive by means of a battery power limiter object, which receives in input the maximum electrical power available from the battery in function of its SOC and current/voltage data and imposes a maximum fraction of it available to the motor. However, to avoid adding a source of difference between the two models, this fraction was set to 1.

6.1.3 Electric Motor

Electric motor is again a map-based object and data are taken from OLT; the choice of a map-based motor is to avoid adding a specific type of electric motor (AC or DC for example) so to allow maximum flexibility of the model for future studies. The following table reports the main data of the motor

ELECTRIC MOTOR MAIN DATA	
Rated Power [kW]	125
Maximum speed [RPM]	16000
Inertia coefficient [kg*m^2]	0.015

Figure 6.5: Electric Motor Parameter for both Simulink and GT-Drive

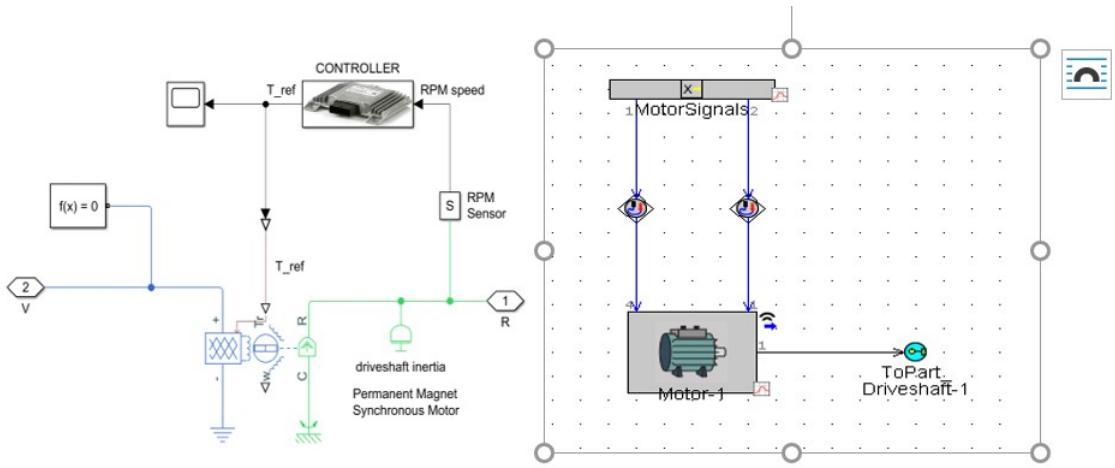

Figure 6.6: Electric Motor models for both Simulink and GT-Drive

- In Simscape the electric motor and the driver are integrated in the same block, while in GT-Drive the motor controller is implemented in another sub-assembly.
- Both models include the inertia of the rotor, Simscape inside the block itself, while in GT-Drive it's added to the driveshaft inertia.
- In Simscape the control mode of the engine is torque-mode control, while in GT-Drive is power-mode control.
- In both models the thermal behaviour has been ignored.
- Both models do not allow intermittent over-torque, maximum value of torque is fixed and limited to the one reported in the maps.
- In both models damping of the rotor is not considered.

6.1.4 Vehicle Body

As mentioned before in the introduction, since the electric motor provided from OLT data was not intended for a pure electric vehicle, and being the electric motor, along with the battery, a key part for the analysis, the choice was to model the vehicle body as a smaller vehicle so to allow the powertrain to propel it properly along the desired driving cycle. In particular, the data were taken from a FIAT Ducato datasheet, so to modify all the factors strictly related to resistant power,

such as vehicle mass, drag coefficient, wheelbase etc. [15] Another important aspect to consider is that in BEV vehicle, no gearbox is needed, so the speed reduction must be done only by the final drive, which will show a higher value with respect to a conventional vehicle final drive.
Comparison of these two models is shown in the next figure:

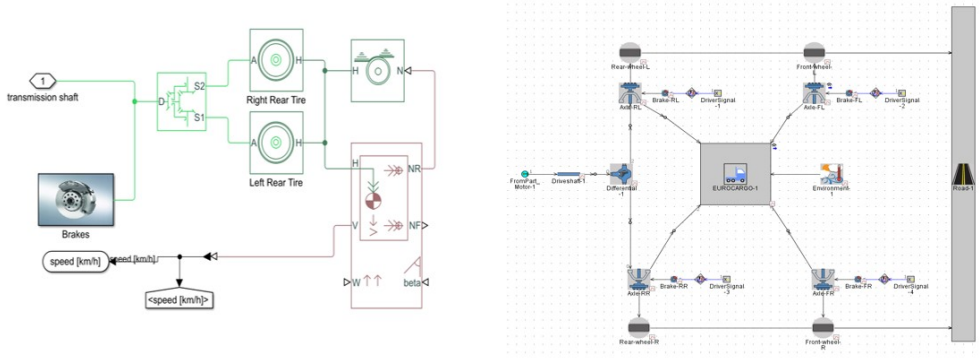

Figure 6.7: Vehicle Body models for both Simulink and GT-Drive

The following table reports all data related to vehicle body:

-	Simulink	GT-Drive
Mass [kg]	3500	3500
Wheel base [m]	3	3
Front axle to COG distance [m]	1.5	1.5
COG height [m]	0.55	0.55
Drag coefficient	0.31	0.31
Frontal area [m^2]	4.5	4.5
Number of wheels	4	4
Wheels diameter [m]	0.4064	0.4064
Wheel inertia [kg*m^2]	2	2
Rolling resistance coefficient	0.008969	0.008969
Final drive	12	12
Driveshaft inertia [kg*m^2]	0.32	0.32

Figure 6.8: Vehicle Body Ducato datasheet for both Simulink and GT-Drive

- In SIMSCAPE environment, driven tyres are not modelled. Rolling resistance is applied to the driving tyres; to take driven tyres into account in the model, an inertia block is added to the driveshaft, with same inertia as driven wheels.
- In GT-DRIVE rolling resistance is applied only on driven tyres.
- In both models, brakes are map-based object which interpolates linearly a braking torque from a maximum value corresponding to full throttle pedal position to 0; maximum braking torque was set to 2000 Nm.
- Driveshaft inertia is applied in SIMSCAPE environment to the driveshaft by means of a specific object, while the electric motor one is specified in the

electric motor block. In GT-drive, instead, motor inertia is added to driveshaft one in a simple shaft object, which is placed between the motor and the differential.

- In both models, inertia of the inner and outer parts of final drive is considered. In SIMSCAPE it is taken into account inserting a "differential block" and specifying them as inertia value of planetary part and gears part. In GT-DRIVE it is instead specified in the differential template.

6.1.5 Longitudinal Driver

Simulations to be run are dynamic ones, so a suitable driver is needed to follow the chosen driving cycle. The two softwares have in them different types of controllers, which show some differences between them but acts similarly.

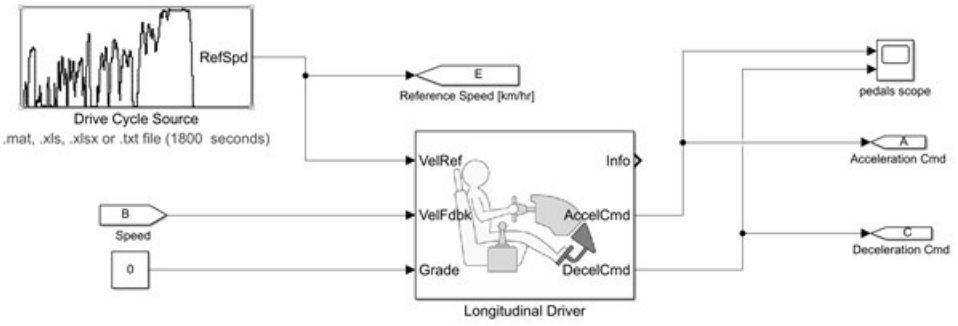

Figure 6.9: Longitudinal Driver Model in Simulink

- Longitudinal driver blockset on SIMSCAPE reads a forward signal of the speed of the vehicle in VelFdbk port, comparing it with the one coming from the reference cycle in VelRef port and giving as output an accelerator and brake commands from 0 to 1.

- in SIMSCAPE the driver is modelled as a PI controller, with the following control parameters :

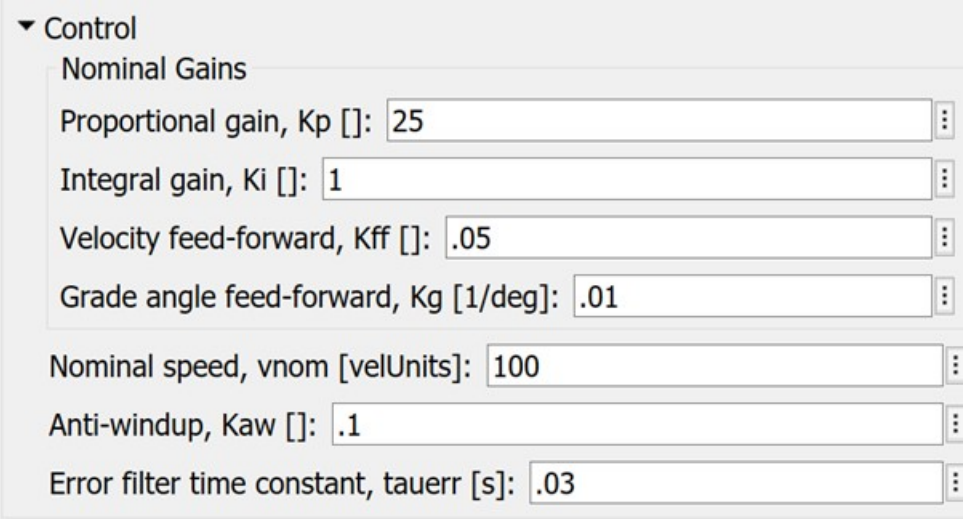

Figure 6.10: Longitudinal Driver Parameters in Simulink

- in SIMSCAPE it is not allowed to give as an output a gear shift command, since in BEV we don't have any gearbox.

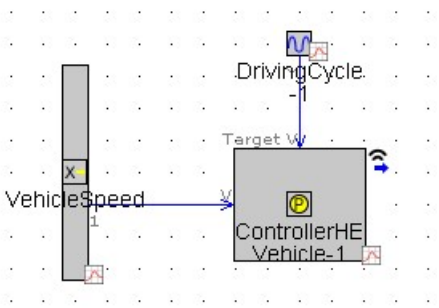

Figure 6.11: Longitudinal Driver in GT-Power

- In GT-Drive, the driver is modelled as a PI controller (in this specific case by means of the object ControllerHEVehicle) which is a feed forward component that senses the actual vehicle speed from vehicle body subassembly sensors and compare it with the imposed driving cycle one, and consequently calculates the requested load, both for traction and braking, needed instant by instant. Load demand is then sent to a Supervisory controller which coordinates all the subsystems of the vehicle.
- In GT-Drive library, this controller is created and intended specifically for the purpose, so no major design is needed, nevertheless a tuning can be done by means of some "aggressiveness factors", analogue in some way to the ones showed for Simscape, but since the driving cycle is followed well from the driver, no modifications of default values have been done.

6.1.6 Comparative Simulation

Once the two models are tuned and set with the same data, it's possible to load a driving cycle and run a simulation; the chosen driving cycle is the WHDC (World

Harmonized Transient Cycle) already used in OLT simulations. SOC is chosen as the main result to be analysed, accordance in SOC trends between the two models will imply same power demand and dynamic behaviour of the vehicle and so the fitting between them.
To compare the results a simulation of the WHDC is run on both environments, then SOC data are exported from both simulations, gathered on Matlab and plotted together to allow a comparison. Results are shown in fig. 6.31.

It results immediately evident how the trends are identical, showing discharge and recharge phases at the same instants.

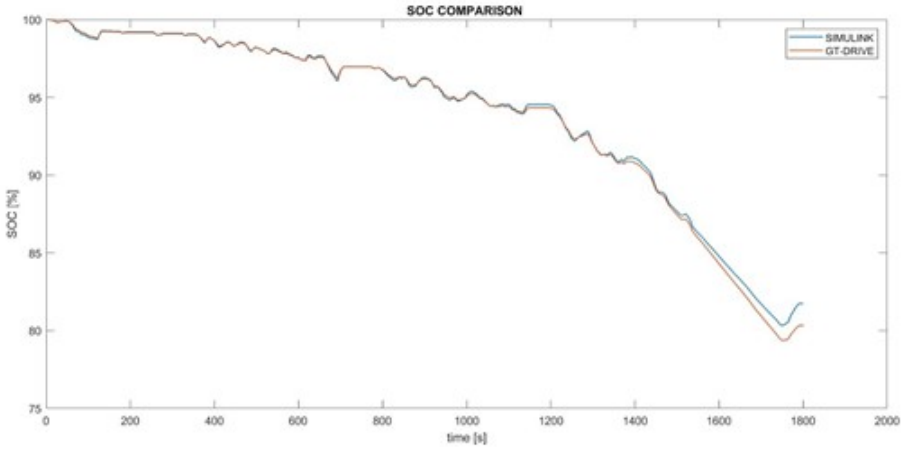

Figure 6.12: SOC comparison on both Simulink and GT-Power

Obviously, following the driving cycle the two curves tend to diverge showing an increasing difference between them due to the different response of the driver, in terms of requested power to accelerate or decelerate the vehicle, which is strictly related to the controller's behaviour. However, the two final values of SOC show a minor difference lower than 0.02, meaning that the two models have comparable results, as shown in the following table:

FINAL SOC VALUE	-
SIMULINK SOC [%]	81.74
GT-DRIVE SOC [%]	80.33
% DEVIATION	1.72%

Figure 6.13: SOC comparison on both Simulink and GT-Power Table

Further Considerations

In order to reduce the gap between the two models, all the parameters of vehicle body, parts and controls have been further analysed. After some trials, the main source of difference was found to be the rolling resistance, which was introduced in the models in slightly different ways, due to intrinsic modeling on the two Simulink and GT-Power software. In order to assess properly the difference, a simulation without rolling resistance is run to show the compliance of the two models.

Results in terms of SOC are reported in the following table:

NO ROLLING RESISTANCE CASE	-
SIMULINK SOC [%]	84.9
GT-DRIVE SOC [%]	85.4
% DEVIATION	0.59%

Figure 6.14: SOC comparison on both Simulink and GT-Power Table Without Rolling Resistance

From new results is even more evident the similarity of the two models, the residual deviation can be attributed to the different driver behaviours, such as time of reaction or aggressiveness. To further increase the compatibility of the two models, further investigations and tuning of rolling resistance could be done so to include it without causing errors, but since the original deviation was considered acceptable, this investigation is not carried out in this work.

6.2 CV

In this section, the conventional Diesel vehicles modelled with Simulink and Gt-Power are compared in order to assess differences and similarities.

6.2.1 Internal Combustion Engine

Internal combustion engine data are the same used in OLT, related to a 4.48 l diesel engines which main characteristics are reported in the following table:

Main Parameters	-
Displacement [L]	4.4853
Inertia coefficient [kg*m^2]	1.44
Idle speed [rpm]	750
Max speed [rpm]	2500
Max torque [Nm]	766
Max power [kW]	119

Figure 6.15: Conventional Vehicle Parameters

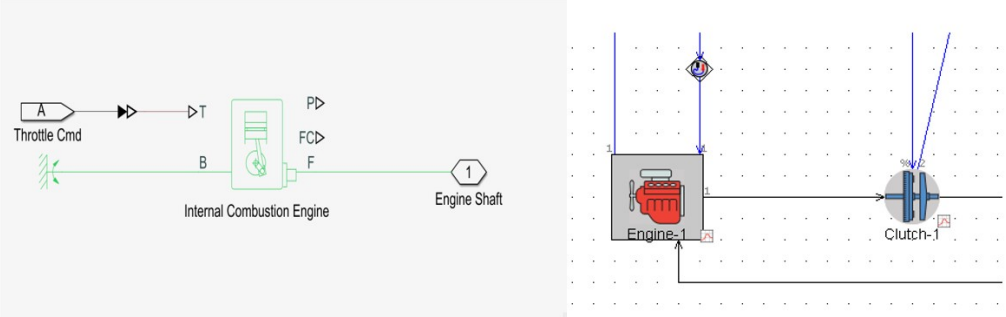

Figure 6.16: Conventional Vehicle Models on both Simulink and GT-Power

- In both SIMSCAPE and GT-Drive environments, internal combustion engine is modelled as a map-based component, maps loaded are:

  1. efficiency map and maximum brake power in SIMSCAPE
  2. engine friction, brake torque and fuel consumption in GT-Drive.

  So the motoring power of the engine is not taken into account in SIMSCAPE model, while it's required in GT-Drive.

- In both models, engine inertia is correctly taken into account in the ICE object.

- Both in SIMSCAPE and GT-Drive environments, ICE is directly controlled by a physical signal related to throttle pedal command coming from the driver, so no conversion in torque or power is necessary.

- Both in SIMSCAPE and GT-Drive environments, an idle speed controller is added so that to avoid the engine speed falls down the idle one during the driving cycle.

- In SIMSCAPE this controller is integrated in the engine block, while in GT-Drive an ECU object is added and placed between the driver and the engine.

6.2.2 Gearbox

For gearbox model generation, data from OLT simulations are taken for the forward speeds, while final drive gear ratio was chosen arbitrarily so to get a good fitting between engine speed and vehicle speed along the driving cycle. Gearbox data are reported in the following table:

Speed	1	2	3	4	5	6	FD
Gear ratio	6.02	3.32	2.07	1.4	1	0.79	2

Figure 6.17: Conventional Vehicle Gearbox Ratio

The two models are shown in the following in 2 separated images in order to allow the good fitting in these sheets.

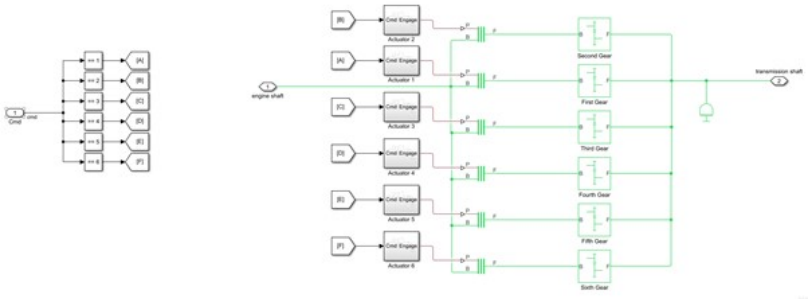

Figure 6.18: Conventional Vehicle Gearbox Model Simulink

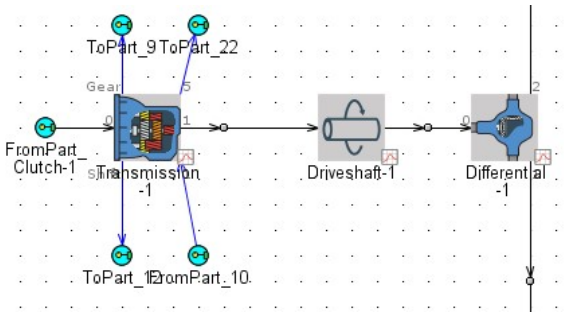

Figure 6.19: Conventional Vehicle Gearbox Model GT-Power

- In SIMSCAPE environment, gearbox is modelled as several power lines in parallel. Each one has a pair gears, characterized by the transmission ratio values specified in OLT data, and a clutch (so 6 clutches are modelled), which is correctly engaged by the related actuator when the shift command is applied. Regarding GT-Drive, instead, a specific object for gearbox and one for clutch are present, where it's possible to specify all the parameters such as gear ratios or inertias.

- In SIMSCAPE and GT-Drive clutch is an ideal model, so it doesn't add damping or smoothing of the engagement. Since no clutch data are provided from OLT model,the choice was to insert reference data from other sources.

- In SIMSCAPE the inertia of the transmission part of the gearbox, as well as of the engine part, is considered adding two inertia blocks, while in GT-drive is specified in the gearbox object itself.

- In SIMSCAPE the inertia of the pair gears is taken into account adding in each power line a proper "inertia block".

6.2.3 Vehicle Body

For what the vehicle body is concerned, components and elements are the same of BEV architecture. Data of Ducato are inserted, and considerations made for BEV comparison hold also in this case. An important difference with the BEV models, however, is the brake system, since in the latter the required braking torque was splitted in two components, one from the electric motor and one from the brakes, in this conventional vehicle model, instead, all the braking torque is provided by the brakes. All data are the same of BEV and so not reported.

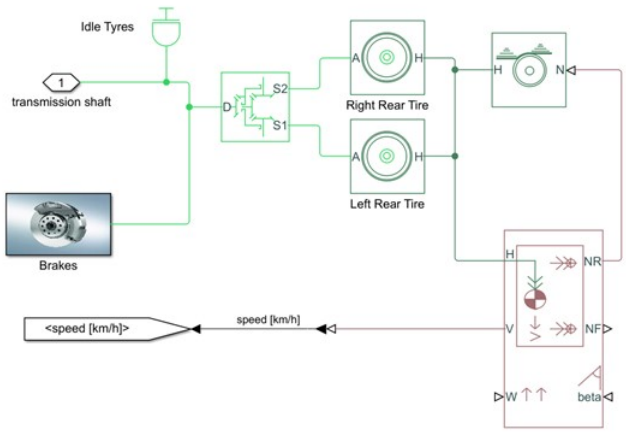

Figure 6.20: Conventional Vehicle Body Simulink

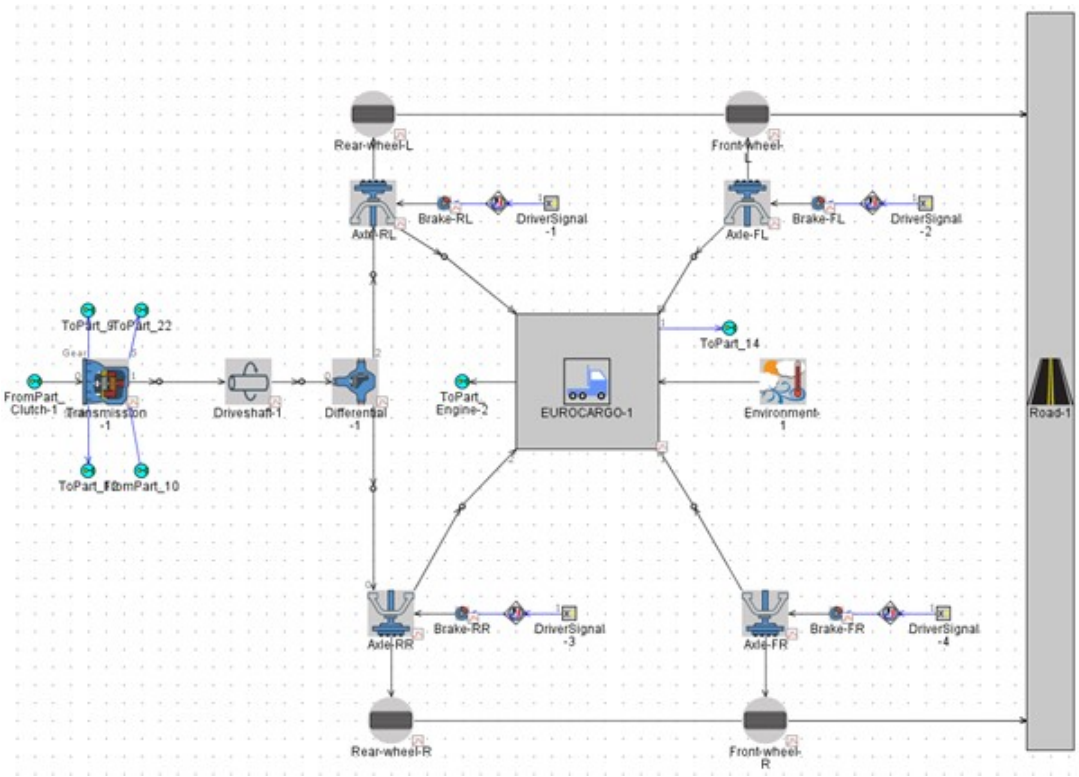

Figure 6.21: Conventional Vehicle Body GT-Power

6.2.4 Longitudinal Driver

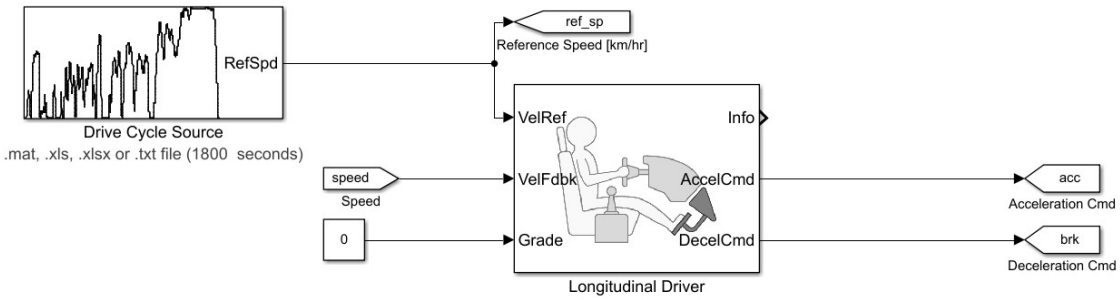

Figure 6.22: Conventional Vehicle Driver Simulink

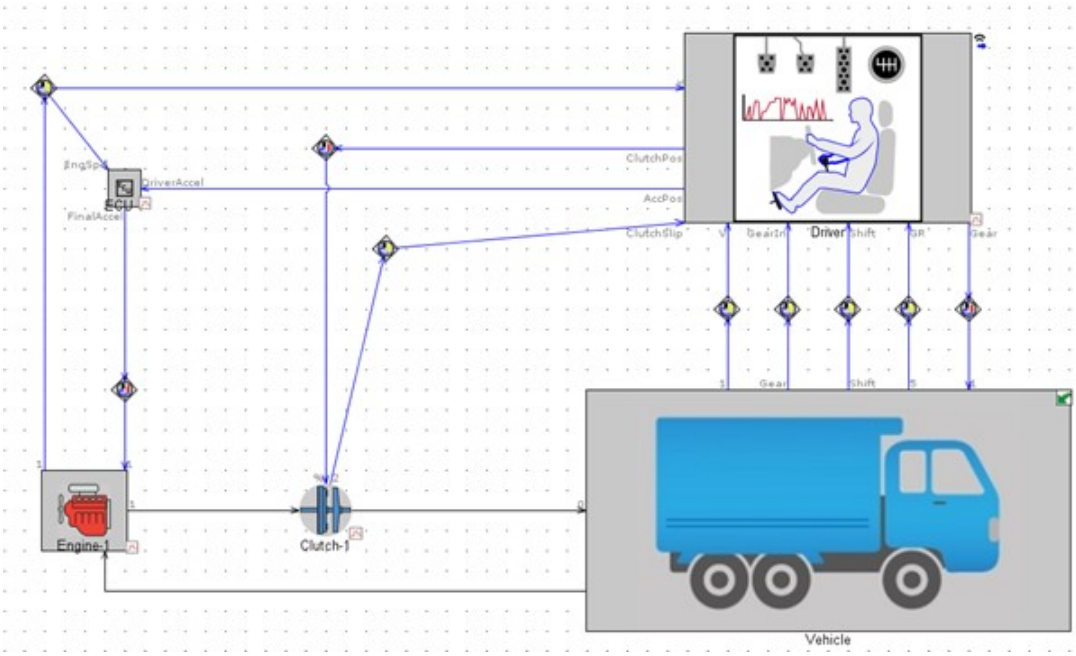

Figure 6.23: Conventional Vehicle Driver GT-Power

Regarding the driver, in SIMSCAPE some modifications are done to allow the driver to follow better the driving cycle, since the presence of gearbox introduces the complexity of gear shifting management, making the system harder to control.

In particular, the proportional gain Kp value is reduced to 5; the new parameters are shown in the following

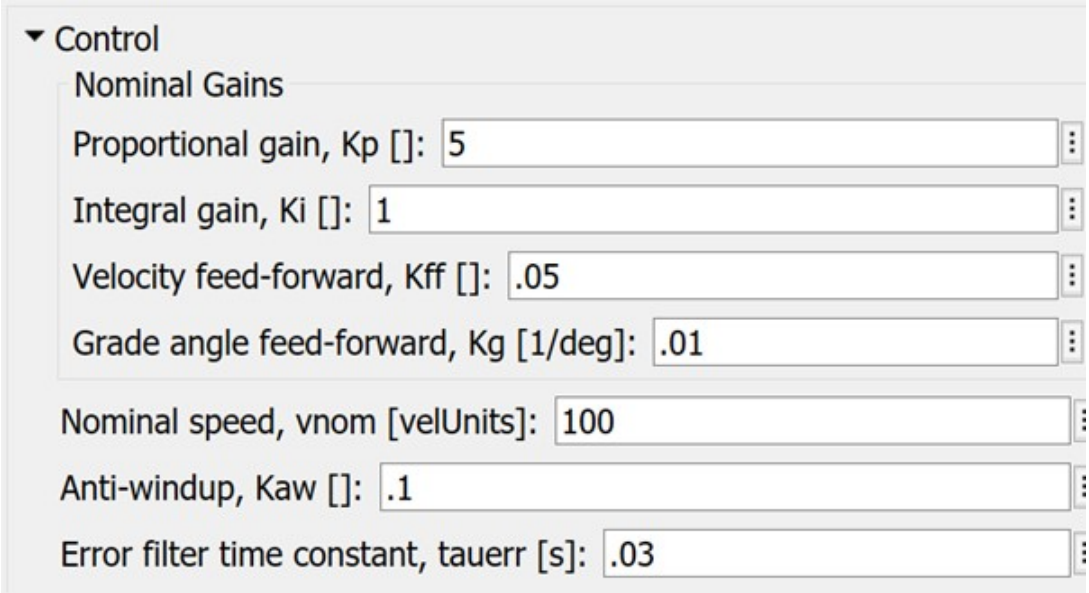

Figure 6.24: Simulink Driver Parameters

Regarding GT-Drive, a different driver object is used, which is still a PI controller such as in BEV model, but also integrates all the functions related to gearshifts and clutch control, which were not necessary in the BEV case. The main functions and parameters are however the same as for BEV case, and also in this case no modification of the default aggressiveness factors values was done.

6.2.5 Gearshift Logic

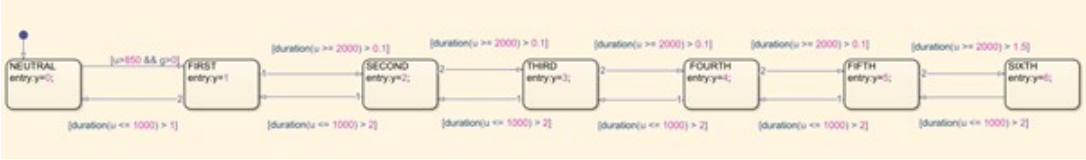

Figure 6.25: Simulink Gearshift Logic

In Simulink the gearshift logic is implemented with the help of Stateflow add-on. In Stateflow upshifting and downshifting logic is based only on engine speed. It means that a threshold of upshifting (2000 rpm) and downshifting (1000 rpm) is established. When the engine speed overcomes the upper threshold, the driver will upshift, while if it goes below the lower threshold for a predefined time (2 seconds), the driver will downshift, this time interval is added to avoid unrequired downshifts immediately after upshifting. Moreover, in the case of upshifting manoeuvres, a reaction delay time of 0.1 second is considered. For the case of passing from neutral to first gear, to avoid speed oscillations, a further condition is imposed: the reference speed must be bigger than 0 to allow this gearshift.
In GT-Drive gearshift logic is implemented directly inside the driver object, where the same criteria described regarding Stateflow are also implemented so to have a comparable logic between the two models. An important feature of the GT-Drive model is the possibility to define a fuel cut-off strategy inside the ECU object. In this

case, however, no cut-off is implemented so to be as close as possible to SIMSCAPE model.

6.2.6 Comparative Simulation

Once the two models are tuned to be as similar as possible, a simulation is run. The chosen driving cycle is the WHVC, the same selected in the BEV comparison and OLT simulations. The first comparison shown is the vehicle speed profile one, comparing the target speed and the actual one in both models, so to see how the models are able to follow the requested one.

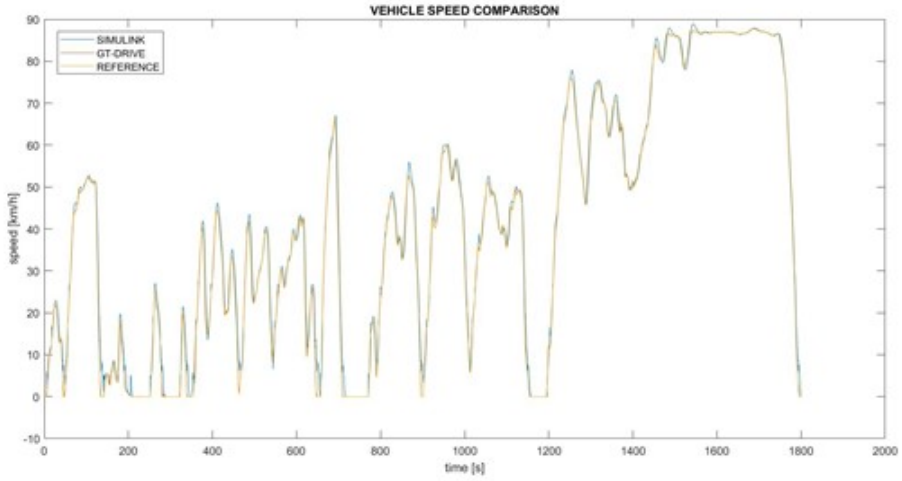

Figure 6.26: Final Simulation Comparison

Looking at the plot, it's possible to notice that both models follow well the driving cycle, some minor differences are however present due to differences in gear shift logic implementation and driver reactions. These differences can be appreciated looking at a magnification of the previous plot, which is reported in the following

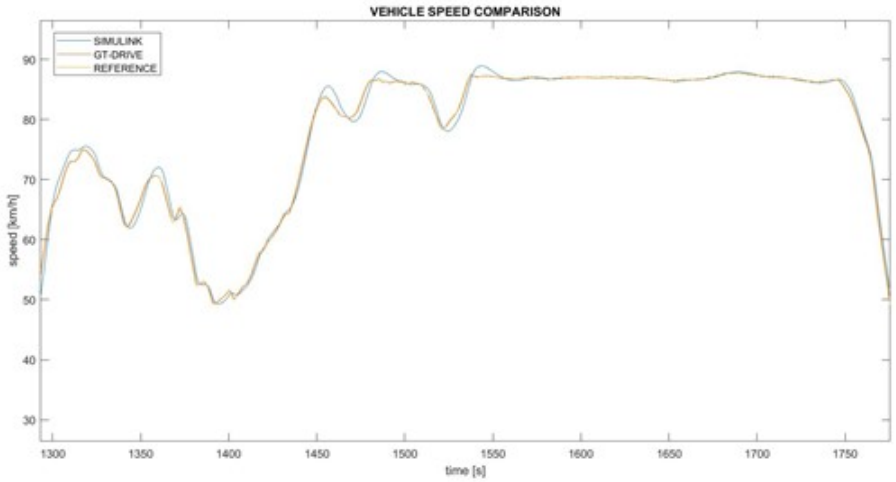

Figure 6.27: Final Simulation: Magnification

From this plot it's possible to see how the two models react differently when the speed profile changes
In particular the Simulink one, during the acceleration phases, always upshifts overcoming a the reference speed. This is due to the imposed time threshold of upshifting, set to 1 tenth of second, to take into account an average reaction time of the driver. During the deceleration phase and the coast phase, the two profiles match the reference one with good precision. For what the gear profile is concerned, the two models' output is plotted in the following

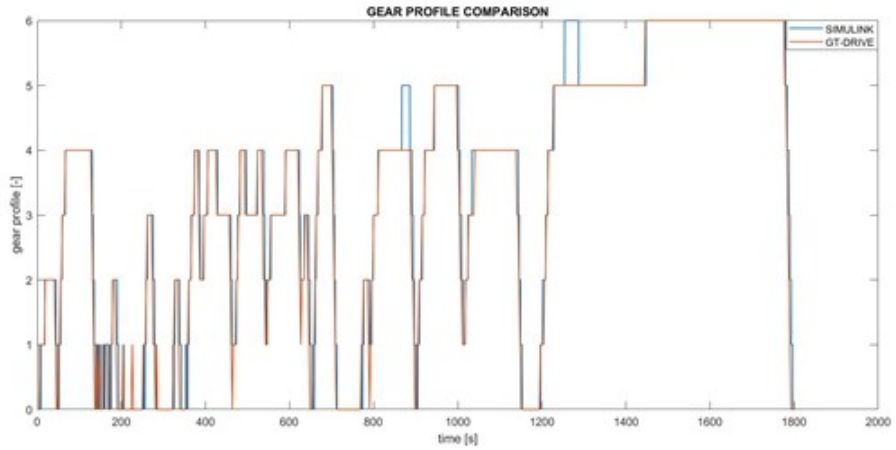

Figure 6.28: Final Simulation: Gearshift Profile

It is possible to note how the gear selection by the drivers during the simulation are almost identical, with the presence of some exception, due to the engine speed oscillation in the two models. It is also confirmed that the GT-DRIVE model driver often upshifts earlier, as explained before. Since the type of simulation run is dynamic, we expected differences of this sort, those differences results evident when looking at the cumulative fuel consumption, which is highly affected by the driver behaviour. Cumulative fuel consumption plot is reported in the following

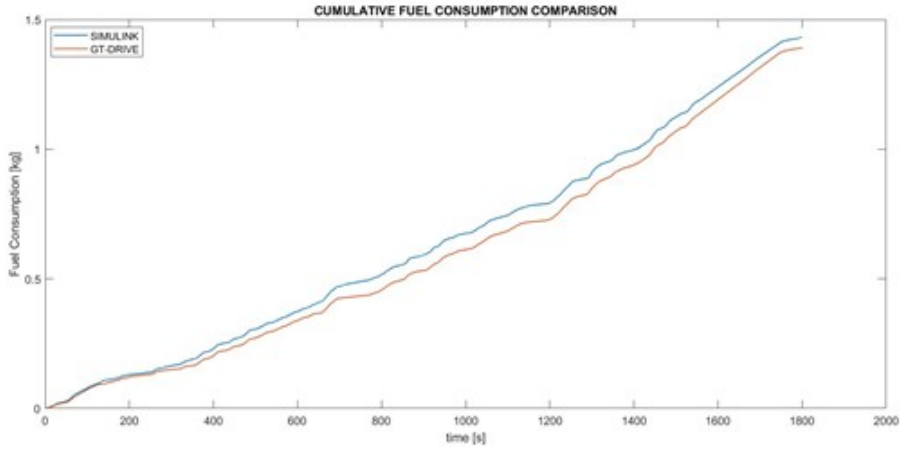

Figure 6.29: Final Simulation: Cumulative Fuel Consumption

Analysing this plot, it's possible to see that the two models have almost the same

trend and similar cumulative fuel consumption value at the end of the cycle. By the way, as the table below shows, a non-negligible deviation value is present:

Fuel consumed [g]	-
SIMULINK	1432
GT-DRIVE	1391
% Deviation	2.86%
Deviation [g/km]	2.05

Figure 6.30: Final Simulation: Results

Further Considerations

Since, as shown in the last paragraph, a deviation between the two values of fuel was present, further investigations about possible reasons are made. Assuming, as widely shown in the BEV comparison chapter, that the rolling resistance model has a non-negligible role in affecting results, looking more thoroughly at the speed profiles, it is possible to note that SIMULINK model driver upshifts with a certain lag with respect to the GT-DRIVE one, as already explained. It brings the engine working outside of the threshold speed for a certain period of time; this could be responsible of the diverging cumulative values of fuel consumption. To confirm this hypothesis, a simulation with a slightly modified gear shift logic in Simulink is run. In this modified version the driver immediately upshifts when the upper speed threshold is reached. Running the simulation, the obtained speed profile is much closer to the reference one, and also to the GT-Drive one. Final cumulative value of fuel consumption is extrapolated new result is compared to the original GT-Drive one in the table shown below:It is evident to note that the percentage deviation is strongly reduced, confirming the hypothesis that driver behaviour strongly influences final results of the simulations. However, since the delay in upshifting was originally added to better simulate real driving conditions, this deviation should not be considered as an error, but just to be imputed to different design choices.

NO REACTION TIME CASE	-
SIMULINK TOTAL FUEL [g]	1382
GT-DRIVE TOTAL FUEL [g]	1391
% DEVIATION	0.65%

Figure 6.31: Final Simulation: No Reaction Time Case Results

It is evident that the percentage deviation is strongly reduced, confirming the hypothesis that driver behaviour strongly influences final results of the simulations. However, since the delay in upshifting was originally added to better simulate real driving conditions, this deviation should not be considered as an error, but just to be imputed to different design choices.

Bibliography

[1] Mathworks. *Simscape Legend of Domains.* https://it.mathworks.com/help/physmod/simscape/ug/domain-specific-line-styles.html, 2020.

[2] Florian Winke. *Transient Effects in Simulations of Hybrid Electric Drivetrains.* 2017.

[3] S Longo G Mohan, F Assadian. *Comparative analysis of forward-facing models vs backward-facing models in powertrain component sizing.* 2013.

[4] Trevor Hastie Robert Tibshirani Gareth James, Daniela Witten. *An Introduction to Statistical Learning.*

[5] Mathworks. *Simscape Driveline.* https://it.mathworks.com/products/simscape-driveline.html, 2020.

[6] Mathworks. *Stateflow.* https://it.mathworks.com/products/stateflow.html, 2020.

[7] Mathworks. *Simscape Electrical.* https://it.mathworks.com/products/simscape-electrical.html, 2020.

[8] Mathworks. *If - Then - Else Blocks.* https://it.mathworks.com/help/simulink/slref/if-then-else-blocks.html, 2020.

[9] Mathworks. *If Action Block Subsystem.* https://it.mathworks.com/help/simulink/slref/ifactionsubsystem.html, 2020.

[10] Mathworks. *Longitudinal Driver Help.* mathworks.com/help/vdynblks/ref/longitudinaldriver.html, 2020.

[11] Mathworks. *Disk Friction Clutch.* mathworks.com/help/physmod/sdl/ref/diskfrictionclutch, 2020.

[12] Mathworks. *Generic Engine.* it.mathworks.com/help/physmod/sdl/ref/genericengine.html, 2020.

[13] Mathworks. *StateFlow Chart.* https://it.mathworks.com/help/stateflow/gs/stateflow-charts, 2020.

[14] Fellow IEEE Cristian Musardo, Giorgio Rizzoni and Benedetto Staccia. *A-ECMS: An Adaptive Algorithm for Hybrid Electric Vehicle Energy Management.*

[15] FIAT. *FIAT Ducato Scheda Tecnica.* www.fiatprofessional.com, 2020.