Project Report
Adaptive Cruise Control

Compliance Design of Automotive Systems
a.y. 2020/2021

Group Members:
Barro Alessandro, Bruscoli Nicolas, Ceccarelli Viviana,
Chiacchiararelli Leonardo, Cintura Manuel, Mariani Lucia

April 16, 2021

Contents

1 Introduction **3**

2 Project organization **4**
2.1 Model based design 4
2.2 Team organization 4

3 Model explanation & design **5**

4 Project development **6**
4.1 Requirements 6
4.2 Unity & Integration testing 6
4.2.1 Switch subsystem 7
4.2.2 Torque split 8
4.2.3 SoC handler 9
4.2.4 Integration test 10
4.3 MIL 10
4.4 SIL 10
4.5 PIL 10

5 Conclusion **12**

List of Figures

4.1 An overview of the controller block 6
4.2 An overview of the switch subsystem 7
4.3 Inputs of the test: acceptable speed is the one set by the user, target speed is the one of the care ahead 7
4.4 Automatiacally generated test reports 8
4.5 An overview of the torque split subsystem 8
4.6 Test inputs (up) Outputs vs baseline (down) 9
4.7 An overview of the SoC handler chart 9
4.8 Input fed to the SoC handler 10
4.9 OTR output vs baseline (up) PID::P output vs baseline (down) 11

Chapter 1

Introduction

Chapter 2

Project organization

2.1 Model based design

2.2 Team organization

Chapter 3

Model explanation & design

The aim of this project is to design an adaptive cruise control, to design it following the MBD we of course needed a plant that models the vehicle longitudinal dynamics. In ***FIGURE*** you can see the whole plant model. To model the longitudinal dynamics of the vehicle we picked from the simscape driveline library the "vehicle longitudinal dynamics" block. We wanted to make some more detailed simulation so we decided to model the driveline system. The vehicle dynamics block accepts as input the long speed of the vehicle, we associated it to the long speed of the CoG of the wheels. The simscape wheels block model the behaviour of a the wheel in terms of inertia, rolling resistance and slip. The angular speed of the wheel is proportional to the torque given as input. The torque is the output of a mapped motor block, of course modifyed by a gear ratio block that models the one-speed transmission of the electric vehicle. When a negative torque is asked, the brake block provides a negative torque at the wheel input. With this plant model we wanted to model the entire driveline from the ECU torque request to the wheels motion.

Chapter 4

Project development

4.1 Requirements

4.2 Unity & Integration testing

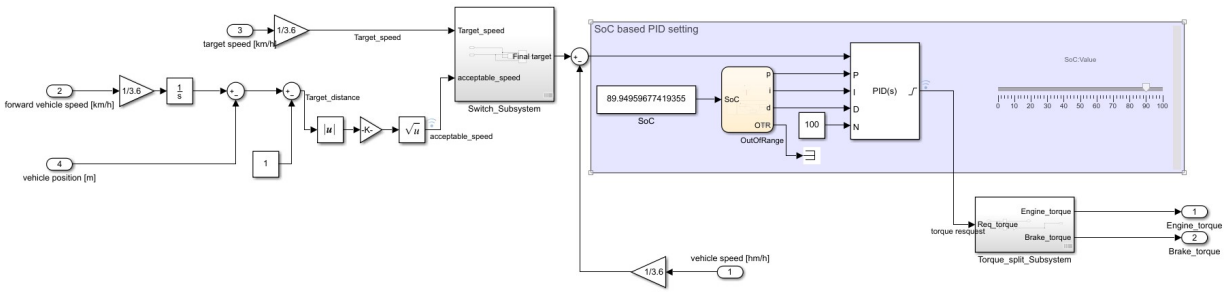

Figure 4.1: An overview of the controller block

Our controller is composed by 4 main blocks: switch_subsys, SoC handler, torque_split and the PID regulator. While it is impossible to test the PID behaviour without a model to be controlled, it is necessary to test separately the other blocks. We tested the units using the tool Simulink Test. We first built an harness for each unity to isolate it from the rest of the model, then we built the input signals in such a way to stimulate the unity in all possible conditions. We also built the baseline signals to tell the tester what output we expected when feeding the unity with a given input.

4.2.1 Switch subsystem

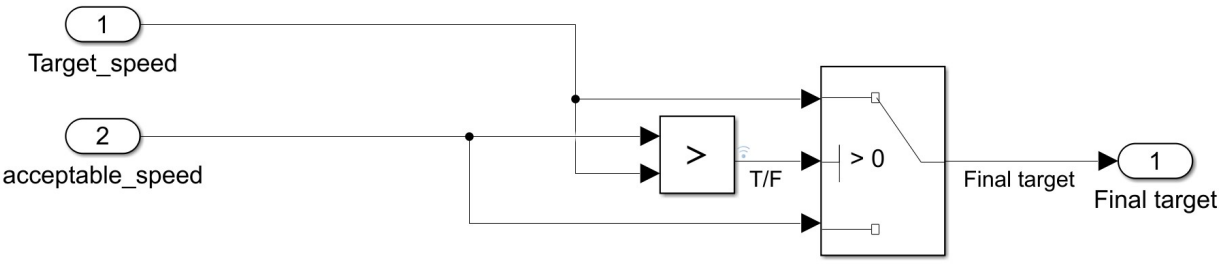

Figure 4.2: An overview of the switch subsystem

This system is composed of a switch commanded by a majority operetor fed with the same signals that have to be switched. We wanted to be shure that this small unity would have behaved as expected. In Fig 4.3 we can see the inputs that we used to test the unit: we wanted to test all the possible conditions for the 'grater than' block. To better visualize wich signal was let pass by the sitch we ttried to differenciate as much as possible the two inputs. As we can see [FIGURE] the simulink test tool is able to generate

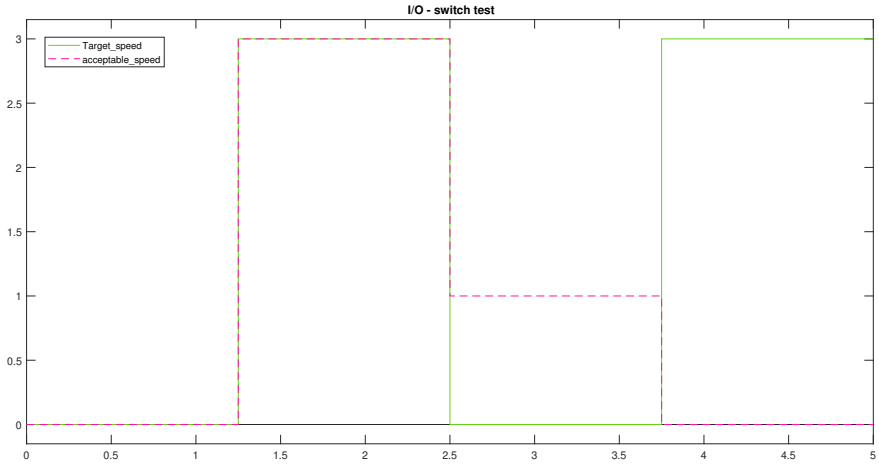

Figure 4.3: Inputs of the test: acceptable speed is the one set by the user, target speed is the one of the care ahead

a document reporting the configuration of the test, to guarantee to the customer (we can call costumer whoever will need to use our subsystem and will need it being tested) that the test was performed accordingly to his needs. This feature also gives the opportunity to an external part to perform the exact same test and verify that the results are coherent. In the same way the tool can generate a report regarding the output of the test, here in Fig. 4.4 we can see that the test produces successful results, infact the output of the test coincides precisely with our baseline signal.

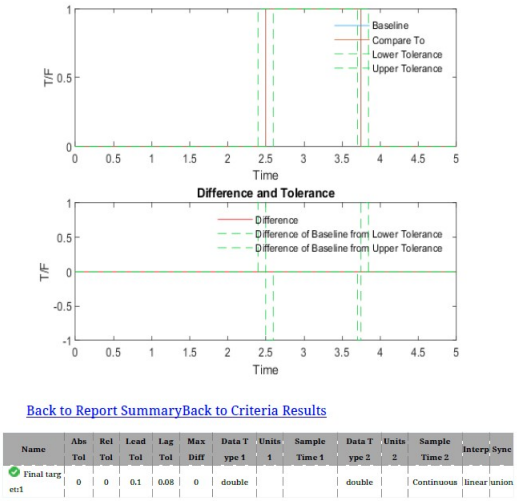

Figure 4.4: Automatiacally generated test reports

4.2.2 Torque split

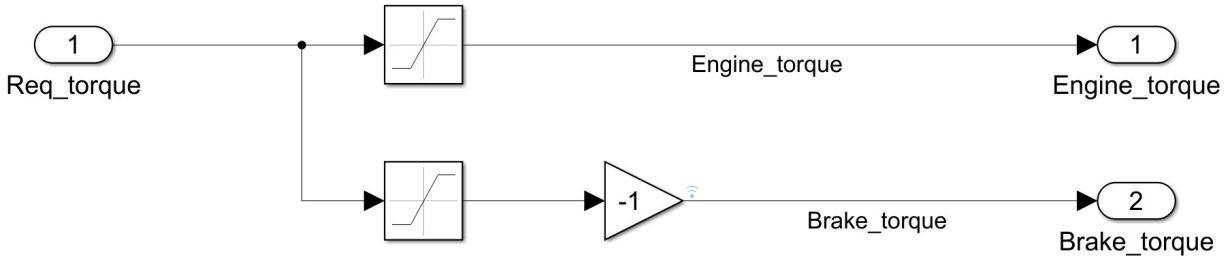

Figure 4.5: An overview of the torque split subsystem

This unit's goal is to split the signal coming from the PID, asking for torque. The PID asks for a negative or positive torque, we have to be able to translate the negative torque in a signal to be sent to the brake module. This unit's goal is to devide the torque signal into two signals depending on the sign of the PID output. To test this unit we just had to verify that when given a positive input it was tranferred to output assigned to the engine and when given a negative input it was transferred to the brake output. At the same time the ouput assigned to the opposite sign has to be zero. In Fig. 4.6 you can see how the test input was designed and you can see that the output corresponds exacly to the baseline signal, meaning a successful test.

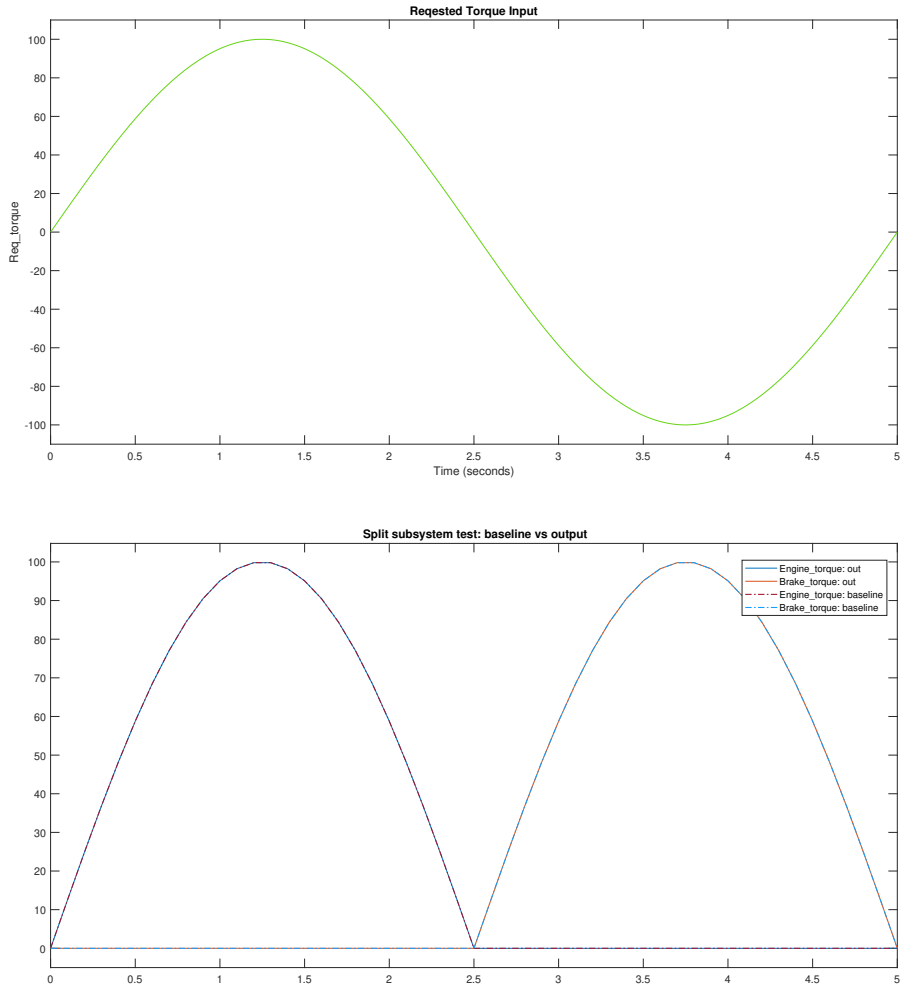

Figure 4.6: Test inputs (up) Outputs vs baseline (down)

4.2.3 SoC handler

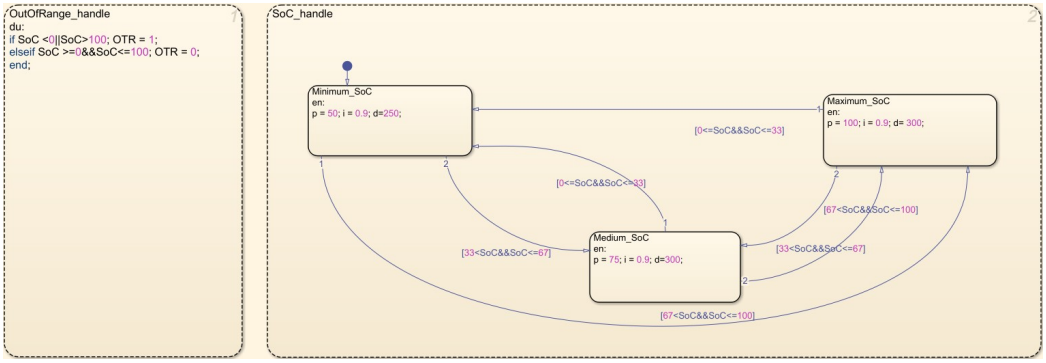

Figure 4.7: An overview of the SoC handler chart

This unit is a FSM, so a slightly more complicated test is due. The goal of this uniti s to manage the aggressiveness of the control depending on the SoC of the vehicle's battery pack. We want a less aggressive control if the car has less autonomy left. We divided the interval of possible SoC values in three segments: minumum, medium, maximum. Each

segment corresponds to a different state. We have three states, since from each state you can go to the other two we have in total 6 transitions. We have only one input: SoC signal. In the input signal, wich you can see in Fig.4.8, we changed the signal in order to stimulate all the possible transitions. To be more robust against unexpected behaviour we wanted an additional output for the FSM. The SoC signal is expected to never trespass the 0% and 100% bounds, so we created a signal "OutOfRange" that rises when this condition is verified. The FSM is designed to check constantly the OTR signal while performing the ordinary tasks, so a parallel state is created in the FSM. We also tested the behaviour

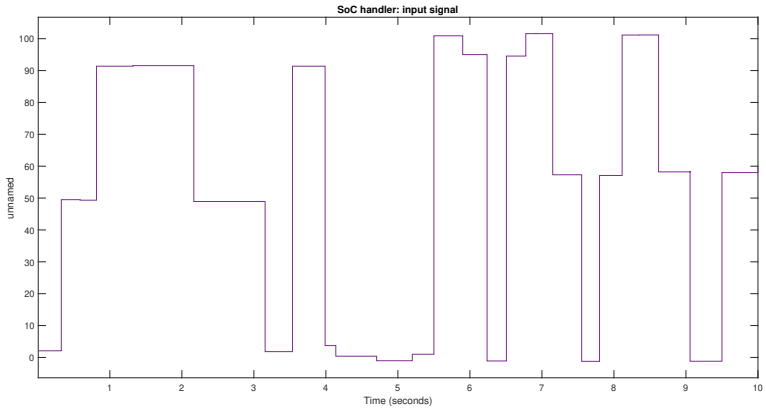

Figure 4.8: Input fed to the SoC handler

of this signal. In Fig.4.8 you can deduce from the input signal (second half of the signal) that we wanted to test that regardless the state in wich we were and the previous value of the SoC signal, the OTR signal would raise when expected to. In Fig.4.9 you can see that the output signal matches the baseline, meaning a succesful test.

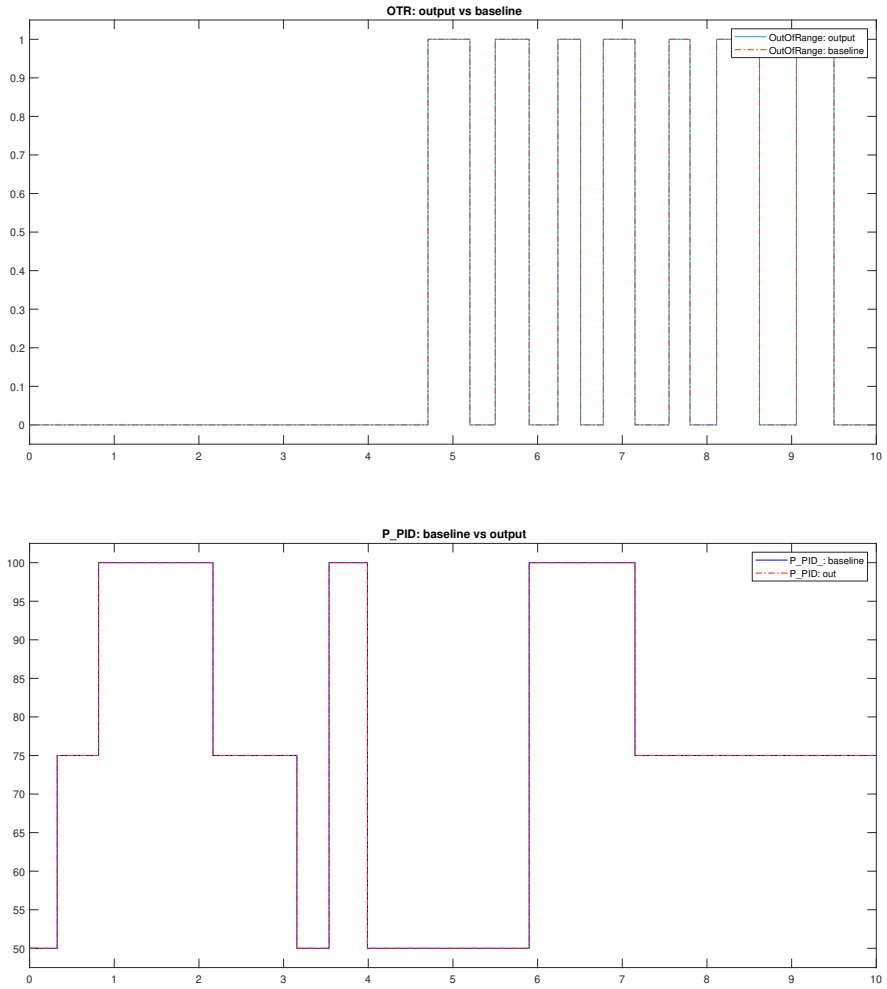

Figure 4.9: OTR output vs baseline (up) PID::P output vs baseline (down)

4.2.4 Integration test

All these units cooperate in the same controller module, so an integration test would be necessary. In our specific case these three units do not have in common any input or output, meaning that there is no possible integration test to be performed. All theese units are meant to condition the PID behaviour, so in order to perform an integration test the PID would have to be involved. Since we assume the PID block itself already test by MATLAB there is no point in testing the PID behaviour when not regulating a plant model. According to the previous affirmations we decided to let the integration test coincide with the MIL test, in order to test the behaviour of the whole controller.

4.3 MIL

4.4 SIL

4.5 PIL

Chapter 5

Conclusion