MIDDLE EAST TECHNICAL UNIVERSITY

Electrical and Electronics Engineering Department

EE498 Control System Design and Simulation Term Project

Adaptive Cruise Control System Using Model Predictive Control

Cem Akıncı - 2093193

Presented to Klaus Werner Schmidt
11.06.2019

Table of Contents

I. INTRODUCTION

This report presents the term project for the EE498 Control System Design and Simulation course. The project focuses on the Model Predictive Control of the Adaptive Cruise Control System. Main purpose of the ACC system is the safe vehicle following at small distances to reduce the fuel consumption and the traffic density. The vehicle plant model which is frequently used in the recent literature is taken as the model of the system. In order to design the MPC for the system, the linear model of the system is discretized first. Afterwards, the MPC is realized in MatLab, implemented and simulated in Simulink.

The MPC realization is explained in detail with the relevant simulation results and all the work done is covered in this report.

II. SYSTEM MODEL, PARAMETERS AND CONSTRAINTS

A vehicle model with Adaptive Cruise Control is illustrated in Figure 1 [1]. For each vehicle i, the length is represented with “L” and the bumper positions are represented with “q”, the velocity is represented with “v”. The distance between each vehicle is formulated as:

$$d = q(i-1) - q(i) - L(i)$$

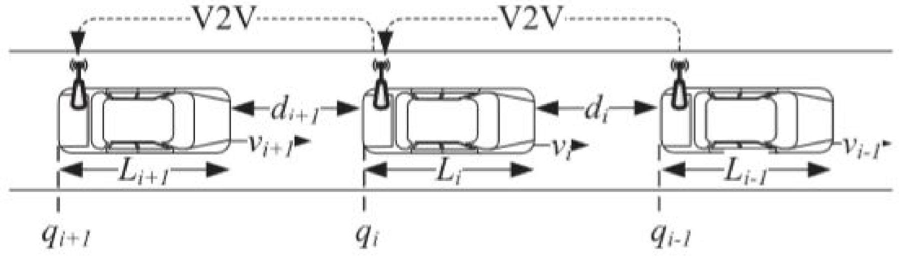


Fig. 1. Vehicle string with CACC.

Figure 1 - A vehicle string using Cooperative Adaptive Cruise Control. [1]

The objective of the ACC system is to regulate the distance between the vehicles such that it converges to a desired small value. A controller capable of satisfying this objective has the following states:

$$\begin{pmatrix} \dot{d}_i \\ \dot{v}_i \\ \dot{a}_i \end{pmatrix} = \begin{pmatrix} v_{i-1} - v_i \\ a_i \\ -\frac{1}{\tau}a_i + \frac{1}{\tau}u_i \end{pmatrix}, \quad i \in S_m \backslash \{1\}.$$

Figure 2 - State Variables of the Cooperative Adaptive Cruise Control Vehicle Model. [2]

The vehicle plant model has the state variables d (distance between the vehicles), v (relative velocity of the vehicle) and a (vehicle acceleration) [2]. The overall plant of the system has the transfer function as illustrated in Figure 3. Here, q(i) represents the vehicle position and u(i) represents the desired acceleration input. The time constant τ is the time constant for the driveline dynamics and the ϕ is the actuator time delay that can differ in each vehicle i. The system is of order three as can be checked through Figure 3. Hence, as expected, there are three state variables of the system.

$$G(s) = \frac{q_i(s)}{u_i(s)} = \frac{1}{s^2(\tau s + 1)} e^{-\phi s}$$

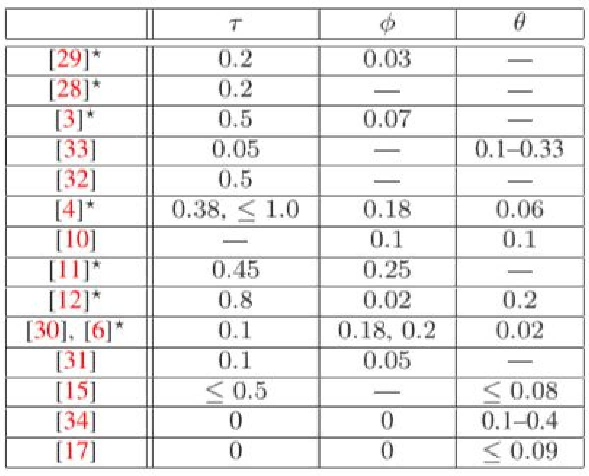
Figure 3 - Linear model of the plant [2].

As mentioned in [1], the respective parameters in Figure 3 are summarized from the practical experiments at Table 1.

Table 1. Parameter values for the linear plant model [1]

TABLE I

PARAMETER VALUES FOR τ_i, ϕ_i AND θ_i



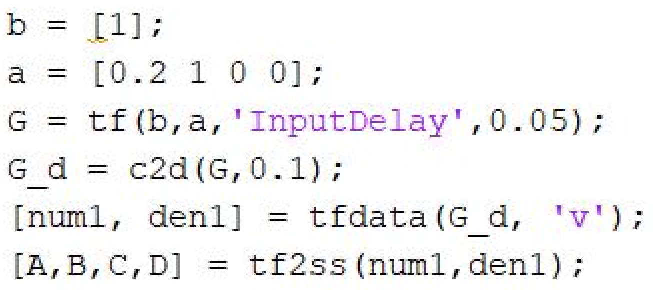
	τ	ϕ	θ
[29]*	0.2	0.03	—
[28]*	0.2	—	—
[3]*	0.5	0.07	—
[33]	0.05	—	0.1–0.33
[32]	0.5	—	—
[4]*	0.38, ≤ 1.0	0.18	0.06
[10]	—	0.1	0.1
[11]*	0.45	0.25	—
[12]*	0.8	0.02	0.2
[30], [6]*	0.1	0.18, 0.2	0.02
[31]	0.1	0.05	—
[15]	≤ 0.5	—	≤ 0.08
[34]	0	0	0.1–0.4
[17]	0	0	≤ 0.09

From the Table 1, the parameter limits are determined such that:

$$0.1 \leq \tau_i \leq 0.8, \quad 0.02 \leq \phi_i \leq 0.25, \quad 0.02 \leq \theta_i \leq 0.2.$$

According to the table, i chose the time constant τ to be 0.2 and the actuator time delay ϕ to be 0.05. The parameter choice is done according to the experiments in the Reference [4].

For the MPC design, the state space model of the given plant model is required. First of all, the linear model is discretized with the "c2d" function in MatLab. The written script is illustrated in Figure 4.



```
b = [1];
a = [0.2 1 0 0];
G = tf(b,a,'InputDelay',0.05);
G_d = c2d(G,0.1);
[num1, den1] = tfdata(G_d, 'v');
[A,B,C,D] = tf2ss(num1,den1);
```

Figure 4 - MatLab script written for the conversion of the transfer function from continuous time to discrete time.

When the script is ran, the following discrete time function and the state space matrices are obtained:

```
                 9.797e-05 z^3 + 0.002002 z^2 + 0.001767 z + 6.734e-05
G_d =  z^(-1) * -----------------------------------------------------
                           z^3 - 2.607 z^2 + 2.213 z - 0.6065

Sample time: 0.1 seconds
Discrete-time transfer function.


A =
   2.6065   -2.2131    0.6065
   1.0000         0         0
        0    1.0000         0
B =
    1
    0
    0
C =
   0.0023    0.0016    0.0001
D =
  9.7969e-05
```

A discrete-time version of the linear model is obtained after this process. The matrices A, B, C and are to be used in the MPC design.

System Constraints and Parameters

There are some constraints to the vehicle and controller parameters related with the ACC system. I tried to find a constraint to the input parameter which is the desired acceleration. From [2], a table for vehicle and controller parameters is obtained.

Table 2. Vehicle and Controller Parameters [2]

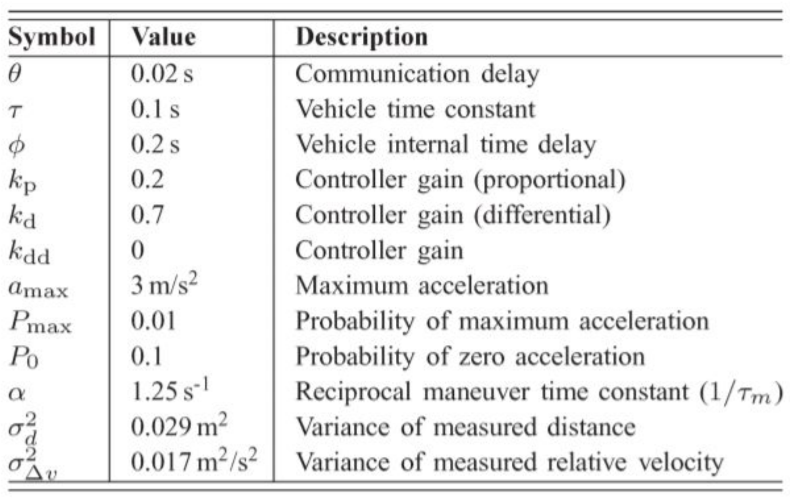


TABLE I
VEHICLE AND CONTROLLER PARAMETERS

Symbol	Value	Description
θ	0.02 s	Communication delay
τ	0.1 s	Vehicle time constant
ϕ	0.2 s	Vehicle internal time delay
k_{p}	0.2	Controller gain (proportional)
k_{d}	0.7	Controller gain (differential)
k_{dd}	0	Controller gain
$a_{\max}$	$3\ \mathrm{m/s^2}$	Maximum acceleration
$P_{\max}$	0.01	Probability of maximum acceleration
P_0	0.1	Probability of zero acceleration
α	$1.25\ \mathrm{s^{-1}}$	Reciprocal maneuver time constant ($1/\tau_m$)
σ_d^2	$0.029\ \mathrm{m^2}$	Variance of measured distance
$\sigma_{\Delta v}^2$	$0.017\ \mathrm{m^2/s^2}$	Variance of measured relative velocity

The time constant and the actuator parameters are different than i chose for my design but it is not important since these parameters differ in each experiment. The parameter limits of time constant and the actuator delay is given in Table 1.
The input constraint for the acceleration can be obtained from Table 2 as a_{max} = 3 m/s^2. The other parameters in the table can be neglected.
The lower limit for the acceleration is obtained from the Reference [3]. The minimum acceleration a_{min} = -5.5 m/s^2 chosen for the ACC system.
Therefore, while simulating the MPC design there will be a constraint on the desired acceleration input not to exceed the predetermined values given above.

At the simulation part, the importance of the input constraint is observed and examined thoroughly.

III. MODEL PREDICTIVE CONTROL REALIZATION IN MATLAB

The discretized model with the given constraints in Part II is built in MatLab and MPC is applied to the model with N=4 Horizon and dual mode prediction. The controller assumes infinite horizon for the values around 4.

The cost matrice Q and the input weight R varies in each application. The cost matrice can be taken either as C^T*C or I depending on the application and the input weight R can be selected between a wide range of values. I simulated the MPC design for different

R and Q combinations in order to see the effect of these parameters. In addition to that, the different R & Q combinations are also examined with different initial conditions and different sampling times.
The initial condition is determined from the experiments in Reference [3]. In Reference [3] it is stated that 40 different experiments are performed for 4 different cases (distance between vehicles is chosen to be 30, 50, 70 and 90 meters respectively).
The initial condition matrix is taken as: x_0 = [60; 40; 0] and the sampling time Ts = 0.1 s.

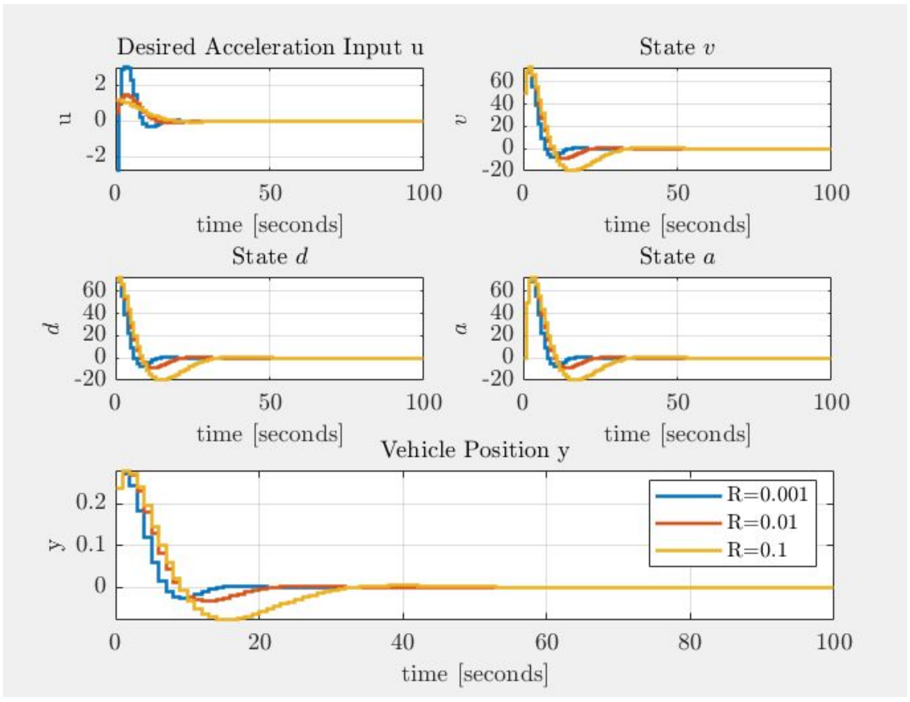


Figure 5 -The states, input and output vs time for different input weights.

From Figure 5, we can deduct that as the input weight R decreases (yellow - red - blue) the input value increases. Hence, the output reaches the steady-state value in a shorter while. It is also important to notice that for the given initial conditions, the input doesn't become unstable even when the input weight is smallest. In that manner, the system performance can be evaluated as good.

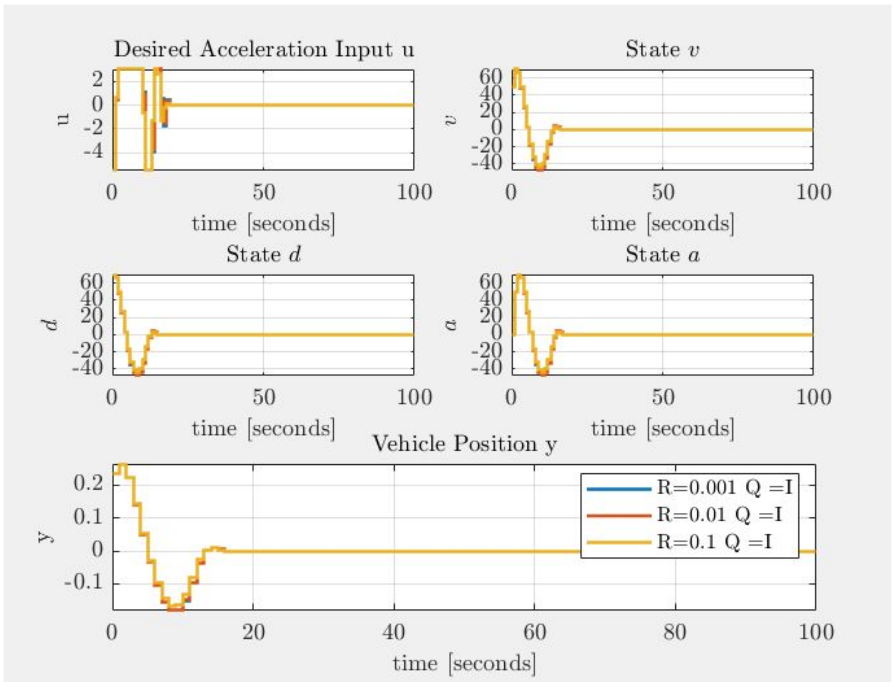


Figure 6 - The states, input and output vs time for different input weights and different cost matrices.

At Figure 6, we can say that changing the cost matrix to identity matrix reduces the effect of the input weight. Obviously, all three cases are very similar. The input saturates for all of the cases when the cost matrix is identity matrix. The differences between the cases would be more clear if the input constraint is removed.

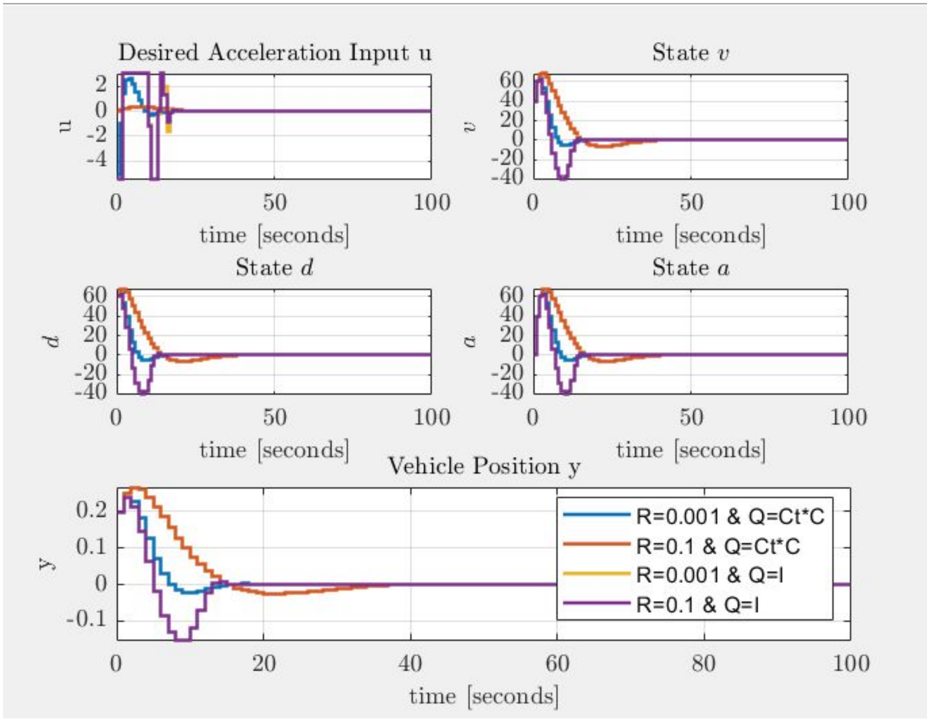


Figure 7 - The states, input and output vs time for different input weight and cost matrix combinations and x_0 = [60; 40; 0].

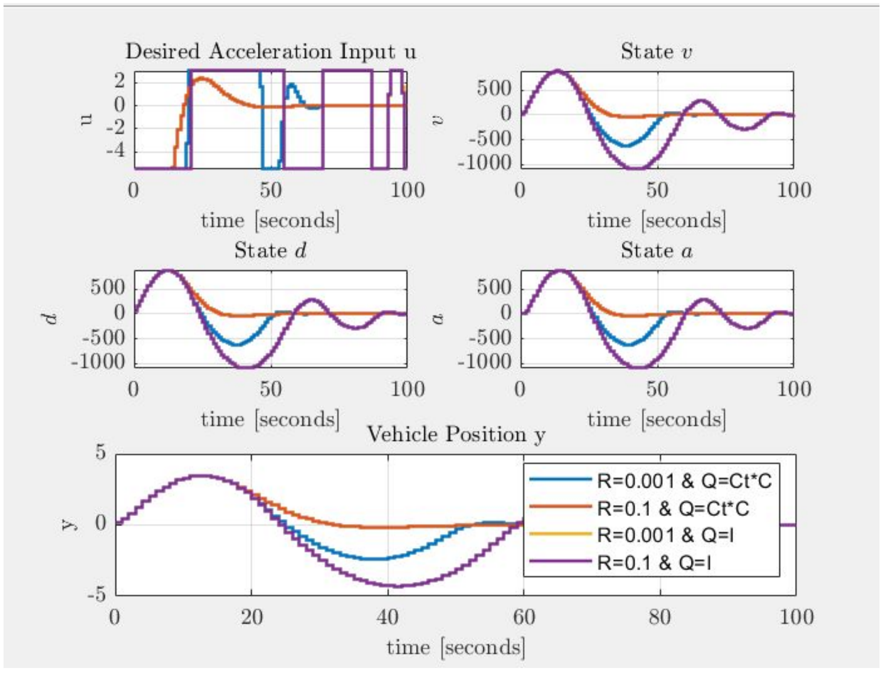


Figure 8 - The states, input and output vs time for different input weight and cost matrix combinations and x_0 = [60; 0; 0].

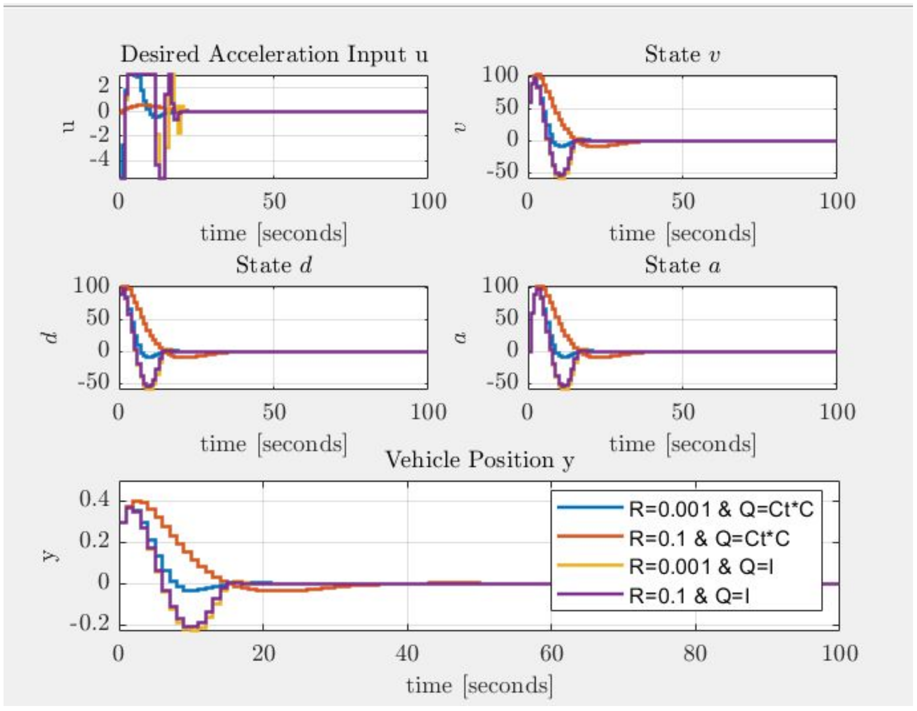


Figure 9 - The states, input and output vs time for different input weight and cost matrix combinations and x_0 = [90; 60; 0].

From Figure 7-9, the simulation is performed with different initial conditions. Looking at the Figures, we can say that when the initial velocity is taken as 0, the input gets bigger and saturates for every case except the case with R=0.1 and Q=Ct*C. The state variables have a great overshoot and as the input is not enough due to the saturation, the state variables oscillate and it takes very long time for these variables to reach their steady state values. The performance of the system is only acceptable when the input weight is high (R=0.1) for this case in terms of stability. However, even in the case illustrated with orange, the states reach to very large values. Thus, it is not suitable to use this initial condition with such an application.

When the initial distance between the vehicles is increased with the initial relative velocity, the response of the system is as expected. Compared with Figure 7, it takes only a little longer for the system to reach steady state. The input and states show similar behaviour in both cases.

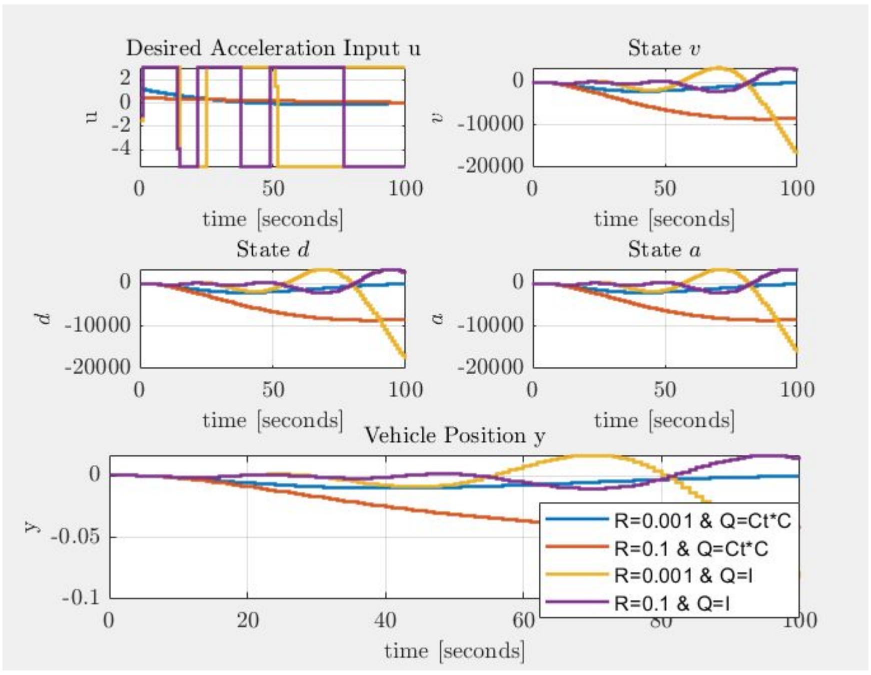


Figure 10 -The states, input and output vs time for different input weight and cost matrix combinations and Ts =0.01 s.

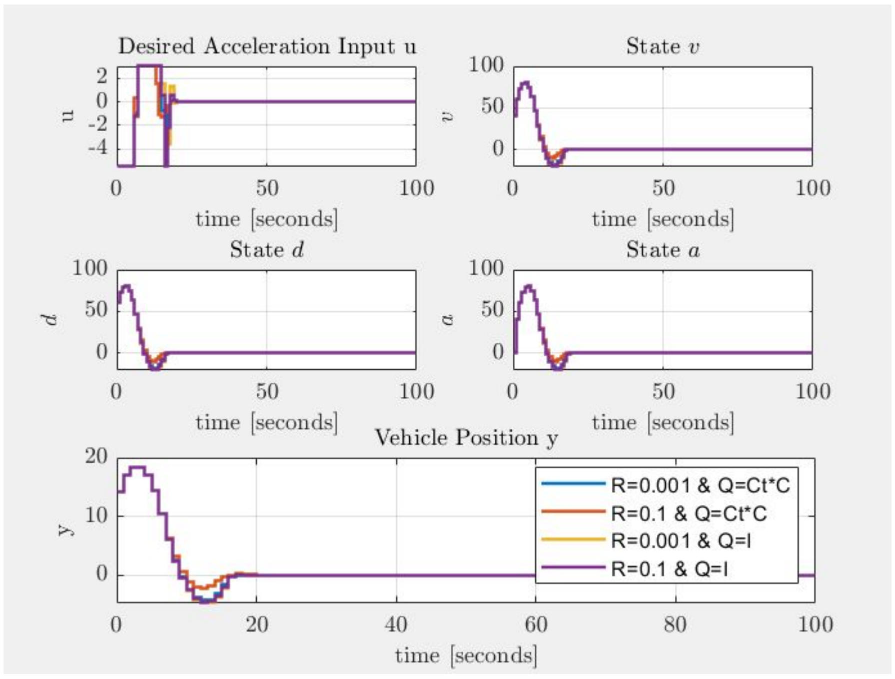


Figure 11 - The states, input and output vs time for different input weight and cost matrix combinations and Ts =0.5 s.

When the Figure 10 and Figure 11 is compared, we can clearly say that if the linear plant model is discretized with a smaller sample time the system show unstable behaviour for all of the cases.

However, when the sample time is increased, the difference between the cases becomes unclear and in all the cases a similar response is observed.

In the previous Figures, we see that sometimes the input is limited to the constraint and can not exceed it. The following simulation is performed with the first initial condition (x_0 = [60; 40; 0]) and with the sampling time of Ts = 0.1 s. The input constraint is changed from [-5.5 , 3] to [-100, 100]. The effect of the input constraint can be observed through the comparison between Figure 7 and Figure 11.

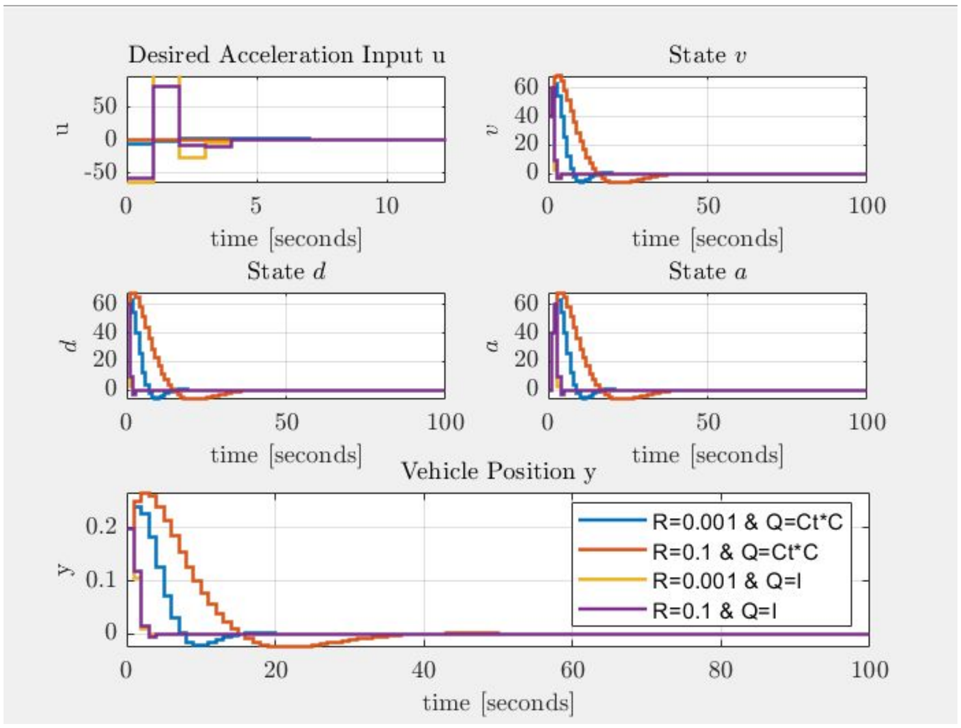


Figure 11 - The states, input and output vs time for different input weight and cost matrix combinations and different input constraints.

In the Figure 11, it can be seen that in some cases the input is still limited to the constraint. However, due to a higher input constraint, the input values are bigger now. Hence, the output reaches to the steady-state value in a very small time especially for the cases colored with purple and yellow. The undershoot observed in the states and the output in Figure 7 is also disappeared in Figure 11 for higher input values. In addition to that, there is still overshoot in the graphs, since the input still saturates.

IV. SIMULINK SIMULATION RESULTS

For the Simulink Simulations, i tried to create my own MPC MatLab function block as implemented in part III, then i tried to simulate the behaviour of a leading car and the controlled vehicle (See Figure 12). The leading vehicle block and the controlled vehicle block is taken from the MathWorks model "mpcACCsystem" [5]. At these blocks, the dynamics between the acceleration and the velocity for both the leading and the controlled vehicle are modeled as :

$$G = 1/s(0.5s + 1)$$

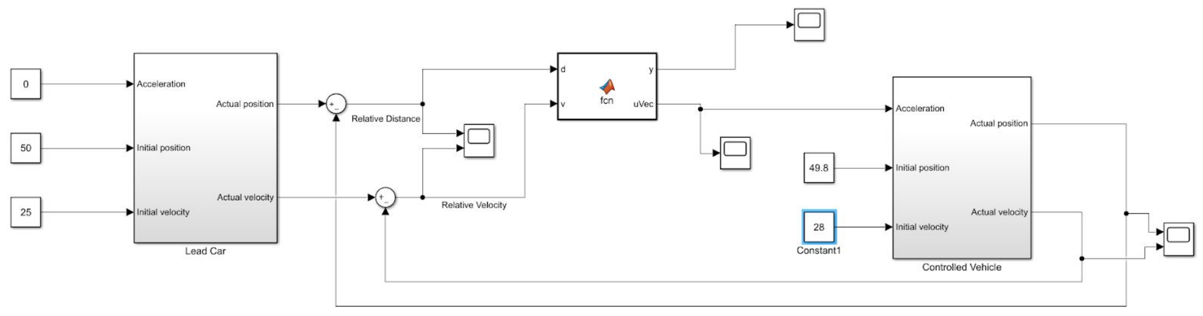


Figure 12 - Proposed method of MPC for the ACC system in Simulink.

The initial position and velocity values are to be set in each vehicle block. I assumed the acceleration of the leading car to be 0. The outputs of the vehicle block are Actual position and Actual velocity, respectively. From these sensed data, the relative distance and relative velocity of the controlled vehicle can be obtained.

The MatLab functiion block takes the relative distance and the velocity as the initial conditions of the controlled vehicle plant. It takes the MPC steps as implemented in part III and produces the output of y (position) and the input for the controlled vehicle which is the desired acceleration (u).

While generating the MatLab function block, i couldn't use the MPC code in Part III directly, since there were too many errors (All of the errors were related to some limitations in the code generation of MatLab.) . Thus, i needed to modify the MPC code such that it operates properly. However, i think i missed something while debugging the errors and the resultant MPC function block didn't operate as the one in Part III.

The problem i couldn't solve is the desired acceleration input is not calculated properly. When the relative velocity and the relative distance is slightly bigger, the input becomes constant at -5.5 and when the relative velocity and distance is slightly lower, the input becomes constant at 3. There are only a few cases when the input takes a value between -5.5 and 3. The following figures illustrate the operation at Simulink:

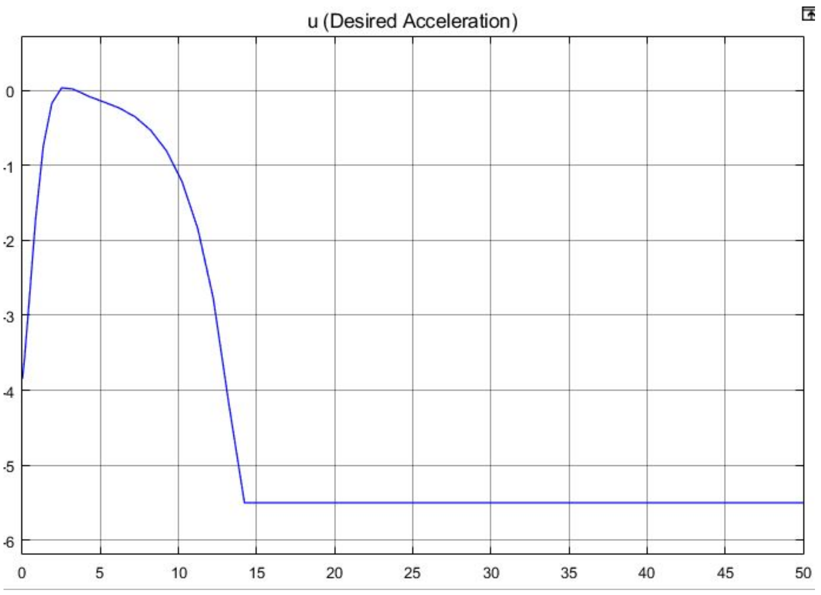


Figure 13 - Desired acceleration input vs time.

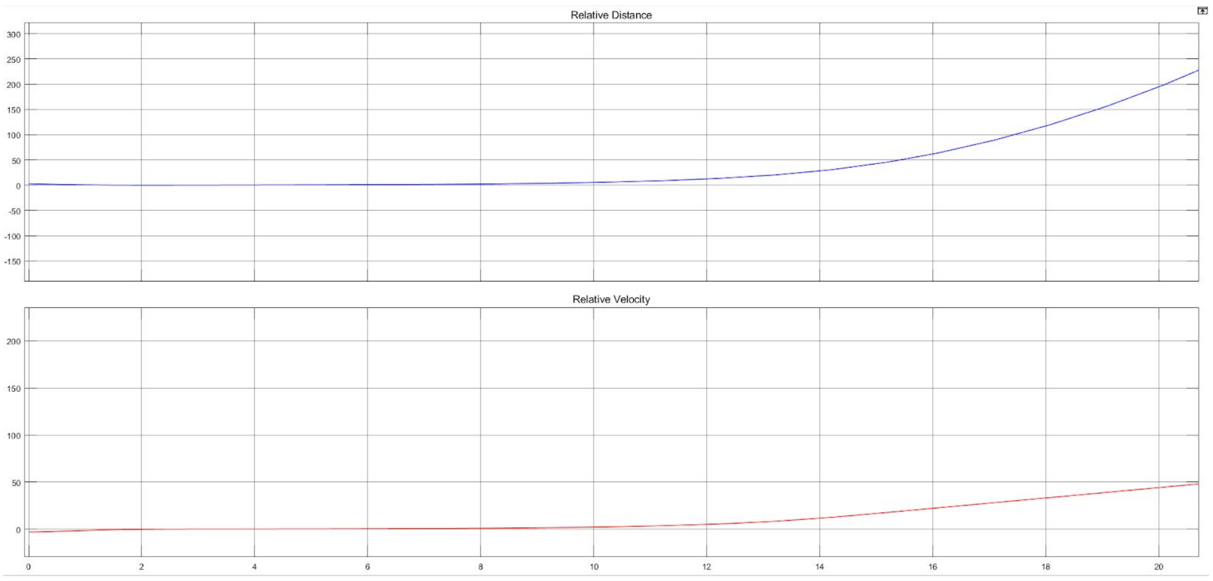


Figure 14 - Relative distance and velocity vs time.

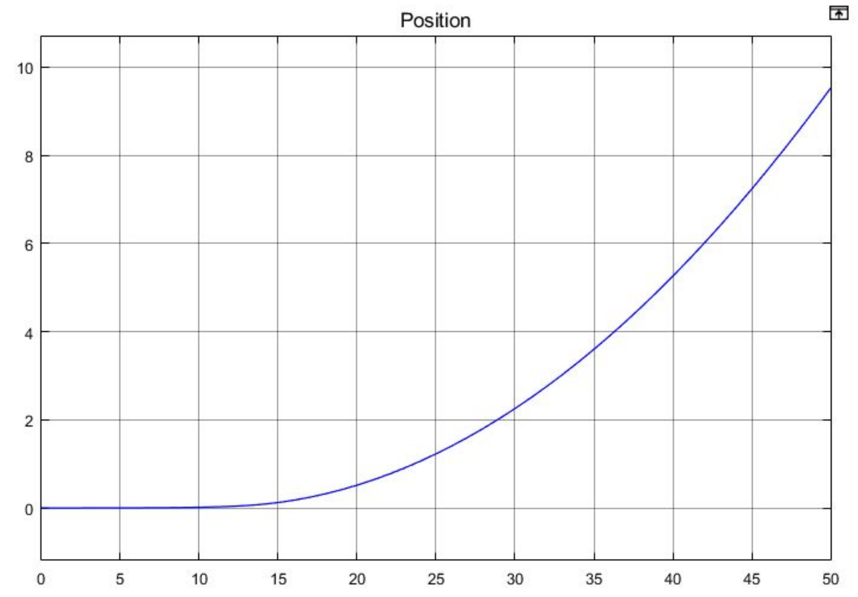


Figure 15 - Position vs time.

At Figure 13 and 14, for an initial relative distance of 3 meters and a relative velocity of 3 m/s, the acceleration, relative distance and velocity vs time is given.

As mentioned above, the acceleration input calculation is somehow problematic in the function block. After T=10, we can see that the vehicle starts to decelerate even it is not necessary and the acceleration input becomes constant at -5.5. The controlled vehicle continues to decelerate and the relative distance and the velocity between the cars gets bigger as time passes.

It can also be verified from the Figure 15 that, the output doesn't converge to zero. Hence, the system is not stable because of the wrong input calculation.

The simulink file is also attached to the other submission documents.

V. CONCLUSION

In this project, a simple MPC design is performed for the Adaptive Cruise Control system. MPC is implemented with different cost matrices and input weights. Moreover, the effect of the initial condition and the sampling time on the MPC behaviour is examined. The operation is conducted in MATLAB.

The project was very beneficial for me, since it made me do some literature search on the topic of ACC and MPC. I made use of the theoretical knowledge covered through the EE498 course. In addition to that, i learned how a basic implementation of MPC using real plant models can be applied in the simulation environment. The finalized design can be improved further in order to have a more safe vehicle following application.

VI. REFERENCES

[1] Al-Jhayyish, A. M. H., Schmidt, K.W: Feedforward Strategies for Cooperative Adaptive Cruise Control in Heterogeneous Vehicle Strings, IEEE Transactions on Intelligent Transportation Systems, Special Issue on "Applications and Systems for Collaborative Driving", vol. 19, no. 1, pp. 113-122, 2018.

[2] J. Ploeg, E. Semsar-Kazerooni, G. Lijster, N. van de Wouw, and H. Nijmeijer, "Graceful degradation of cooperative adaptive cruise control," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 1, pp. 488–497, Feb. 2015.

[3] Luo, Li-hua & Liu, Hong & Li, Ping & Wang, Hui. (2010). Model predictive control for adaptive cruise control with multi-objectives: Comfort, fuel-economy, safety and car-following. Journal of Zhejiang University SCIENCE A. 11. 191-201. 10.1631/jzus.A0900374.

[4] S. Huang and W. Ren, "Longitudinal control with time delay in platooning," IEE Proc.-Control Theory Appl., vol. 145, no. 2, pp. 211–217, Mar. 1998.

[5]
https://www.mathworks.com/help/mpc/ug/adaptive-cruise-control-using-model-predictive-controller.html

VII. APPENDIX

```
% Some figure formatting
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
clear all

numx = [1];
numy = [0.2 1 0 0];
Gcont = tf(numx,numy,'InputDelay',0.05);
%[num den] = tfdata(G, 'v');
G_d = c2d(Gcont,0.1)
[num1 den1] = tfdata(G_d, 'v');
[A,B,C,D] = tf2ss(num1,den1)
x0 = [60;40;0];
Ts = 0.1;
% Optimal control solution for $N = 4$
I = eye(3);
a=A*B;
b=A*A*B;
c=A*A*A*B;
G1 = [0 0 0 0 ; 0 0 0 0 ; 0 0 0 0];
G2 = [B(1) 0 0 0 ; B(2) 0 0 0 ; B(3) 0 0 0];
G3 = [a(1) B(1) 0 0 ; a(2) B(2) 0 0 ; a(3) B(3) 0 0];
G4 = [b(1) a(1) B(1) 0 ; b(2) a(2) B(2) 0;b(3) a(3) B(3) 0];
G5 = [c(1) b(1) a(1) B(1);c(2) b(2) a(2) B(2);c(3) b(3) a(3) B(3)];
G = [G1;G2;G3;G4;G5];
H = [I;A;A^2;A^3;A^4];
Q = C'*C;
R = 0.001;
% Q = eye(2);
Pinf = dare(A,B,Q,R,zeros(3,1),eye(3) );
Kinf = inv(R+B'*Pinf*B)*B'*Pinf*A;
% A*X*A' - X + Q = 0; X = dlyap(A,Q)
P = dlyap( (A-B*Kinf)',Q+Kinf'*R*Kinf);
Qf = P;
Qbar = blkdiag(Q,Q,Q,Q,Qf);
```

```
Rbar = blkdiag(R,R,R,R);
M = G'*Qbar*G + Rbar;
% input bound: umin <= u <= umax
umin = -5.5;
umax = 3;
lb = [umin;umin;umin;umin];
ub = [umax;umax;umax;umax];
% Apply MPC steps
xVec(:,1) = x0;
yVec(1) = C*x0;
uVec = [];
for kk = 1:100
 alpha = G'*Qbar*H*xVec(:,kk);
 Usol = quadprog(M,alpha',[],[],[],[],lb,ub);
 uVec(kk) = Usol(1);
 xVec(:,kk+1) = A*xVec(:,kk) + B*uVec(kk);
 yVec(kk+1) = C*xVec(:,kk+1);
 Xsol(:,1) = xVec(:,kk);
 Xsol(:,2) = A*Xsol(:,1) + B*Usol(1);
 Xsol(:,3) = A*Xsol(:,2) + B*Usol(2);
 Xsol(:,4) = A*Xsol(:,3) + B*Usol(3);
 Ysol(1) = C*Xsol(:,1);
 Ysol(2) = C*Xsol(:,2);
 Ysol(3) = C*Xsol(:,3);
 Ysol(4) = C*Xsol(:,4);
end
uVec = [uVec uVec(end)];
tVec = [0:1:100];

figure;
subplot(3,2,1)
stairs(tVec,uVec,'LineWidth',1.5);
xlabel('time [seconds]')
grid on;
ylabel('u')
title('Desired Acceleration Input u')
ylim('auto')
hold on

subplot(3,2,4)
stairs(tVec,[0 0 1]*xVec,'LineWidth',1.5)
grid on;
xlabel('time [seconds]')
ylabel('$a$')
title('State $a$')
ylim('auto')
hold on

subplot(3,2,3)
stairs(tVec,[1 0 0]*xVec,'LineWidth',1.5)
grid on;
xlabel('time [seconds]')
ylabel('$d$')
title('State $d$')
ylim('auto')
hold on

subplot(3,2,2)
stairs(tVec,[0 1 0]*xVec,'LineWidth',1.5)
grid on;
xlabel('time [seconds]')
ylabel('$v$')
title('State $v$')
ylim('auto')
hold on

subplot(3,2,[5,6])
stairs(tVec,C*xVec,'LineWidth',1.5)
grid on;
xlabel('time [seconds]')
ylabel('y')
title('Vehicle Position y')
ylim('auto')
hold on

set(findall(gcf,'Type','line'),'LineWidth',1.5)
set(findall(gcf,'-property','FontSize'),'FontSize',11);
```

Figure 16 – The MatLab code for the discrete time model with the MPC realization.