Ecole Polytechnique Fédérale de Lausanne

MULTIVARIABLE CONTROL AND COORDINATION SYSTEMS (EE-477)

Case Study: Path tracking for an automated vehicle

Authors:
PEETERS Thomas 288239

January 15, 2021

Contents

1 Introduction

The aim of this case study is to design different models for a path tracking problem. In a first time we will compare 3 models : discrete time linear model, continuous time linear model and continuous time non linear model. In a second time we will design a closed loop model (LQR + controller) and we will tune different parameters to optimize our model.

2 Session 0 : Modeling, linearization, and discretization

Rewrite the non-linear model in the standard form $\dot{x}$ = f(x, u).

If we assume constant curvature $\kappa s = \kappa$ we find :

$$\dot{x_1} = \frac{x_4 cos(x_3)}{1 - x_2\kappa} \tag{1}$$

$$\dot{x_2} = x_4 sin(x_3) \tag{2}$$

$$\dot{x_3} = \frac{x_4}{L} tan(\frac{x_5}{16}) - \frac{x_4 cos(x_3)}{1 - x_2\kappa} \tag{3}$$

$$\dot{x_4} = \sigma_v(v_{ref} - x_4) \tag{4}$$

$$\dot{x_5} = \sigma_\phi(\phi_{ref} - \phi) \tag{5}$$

with x= $\begin{bmatrix} s \\ d \\ \theta_e \\ v \\ \phi \end{bmatrix}$.

Given the chosen state space, why it wouldn't be appropriate linearizing the system around a nominal point ?

A nominal point does not evolve over time and follow the condition $\dot{\bar{x}}(t) = 0$. The approximation for $\dot{x}$ using linearization is close to the exact values when the states are close to the nominal point but the error in the approximation increase when we get away from the nominal point.
In our case, we want to follow a path and we will move away from this operating point and the error in the estimation of $\dot{x}$ will increase. It is therefore suitable to linearize our system around a nominal trajectory.

Calculate the nominal trajectory representing the constant-speed perfect tracking situation (that is, a situation where the vehicle drives perfectly on the path and at a constant speed v_{ref}) for a path of constant curvature κ_{ref} .

We get:

$$\bar{x_1}(t) = v_{ref}.t$$
$$\bar{x_2}(t) = 0$$
$$\bar{x_3}(t) = 0$$
$$\bar{x_4}(t) = v_{ref}$$
$$\bar{x_5}(t) = 16.atan(\kappa.L)$$

Linearize the system around such a nominal trajectory.

To linearize the system around such a nominal trajectory we must calculate the jacobian matrix $\frac{\delta}{\delta x} f(\bar{x}, \bar{u})$ and $\frac{\delta}{\delta u} f(\bar{x}, \bar{u})$:

we find :

$$\frac{\delta}{\delta x}f(\bar{x},\bar{u}) = \begin{bmatrix} 0 & \frac{\kappa\bar{x_4}cos(\bar{x_3})}{(\kappa\bar{x_2}-1)^2} & \frac{\bar{x_4}sin(\bar{x_3})}{\kappa\bar{x_2}-1} & \frac{-cos(\bar{x_3})}{\kappa\bar{x_2}-1} & 0 \\ 0 & 0 & \bar{x_4}cos(\bar{x_3}) & sin(\bar{x_3}) & 0 \\ 0 & \frac{-\kappa 2\bar{x_4}cos(\bar{x_3})}{(\kappa\bar{x_2}-1)^2} & -\frac{\kappa\bar{x_4}sin(\bar{x_3})}{\kappa\bar{x_2}-1} & \frac{tan(\bar{x5}/16)}{L}+\frac{\kappa cos(\bar{x_3})}{\kappa\bar{x_2}-1} & \frac{\bar{x_4}(tan(\bar{x_5}/16)^2/16+1/16}{L} \\ 0 & 0 & 0 & -\sigma_v & 0 \\ 0 & 0 & 0 & 0 & -\sigma_\phi \end{bmatrix}$$

$$\frac{\delta}{\delta u}f(\bar{x},\bar{u}) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \sigma_v & 0 \\ 0 & \sigma_\phi \end{bmatrix}$$

And our linarized equation is equal to :

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}$$

$$A = \frac{\delta}{\delta x}f(\bar{x},\bar{u})$$
$$B = \frac{\delta}{\delta u}f(\bar{x},\bar{u}$$
$$\tilde{x} = x - \bar{x}$$
$$\tilde{u} = u - \bar{u}$$

Discretize the system using Euler approximation
Using the Euler approximation we have :

$$\Phi = I + hA \tag{6}$$
$$\Gamma = hB \tag{7}$$

with h the integrated step.

3 Session 1 : Model implementation and simulation

1.1 Provide the arrays and obtained using the Euler approximation method and the Matlab command c2d.
If we have a sampling time equal to 0.1 second, using the Euler approximation method and the c2d command we find :

$$\Phi_{Euler} = \begin{bmatrix} 1 & 5\text{e-}11 & 0 & 0.1 & 0 \\ 0 & 1 & 0.5 & 0 & 0 \\ 0 & -5\text{e-}21 & 1 & 0 & 0.0078 \\ 0 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0 & 0.500 \end{bmatrix}$$

$$\Gamma_{Euler} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.100 & 0 \\ 0 & 0.500 \end{bmatrix}$$

$$\Phi_{c2d} = \begin{bmatrix} 1 & 5.00e-11 & 1.250e-11 & 0.0952 & 2.886e-14 \\ 0 & 1 & 0.500 & 0 & 0.0017 \\ 0 & -5.00e-21 & 1 & 0 & 0.0061 \\ 0 & 0 & 0 & 0.9048 & 0 \\ 0 & 0 & 0 & 0 & 0.6065 \end{bmatrix}$$

$$\Gamma_{c2d} = \begin{bmatrix} 0.0048 & 0 \\ 0 & 0.0003 \\ 0 & 0.0017 \\ 0.0952 & 0 \\ 0 & 0.3935 \end{bmatrix}$$

1.2 Provide a screenshot of your implementation of the Non linear model - continuous time block.

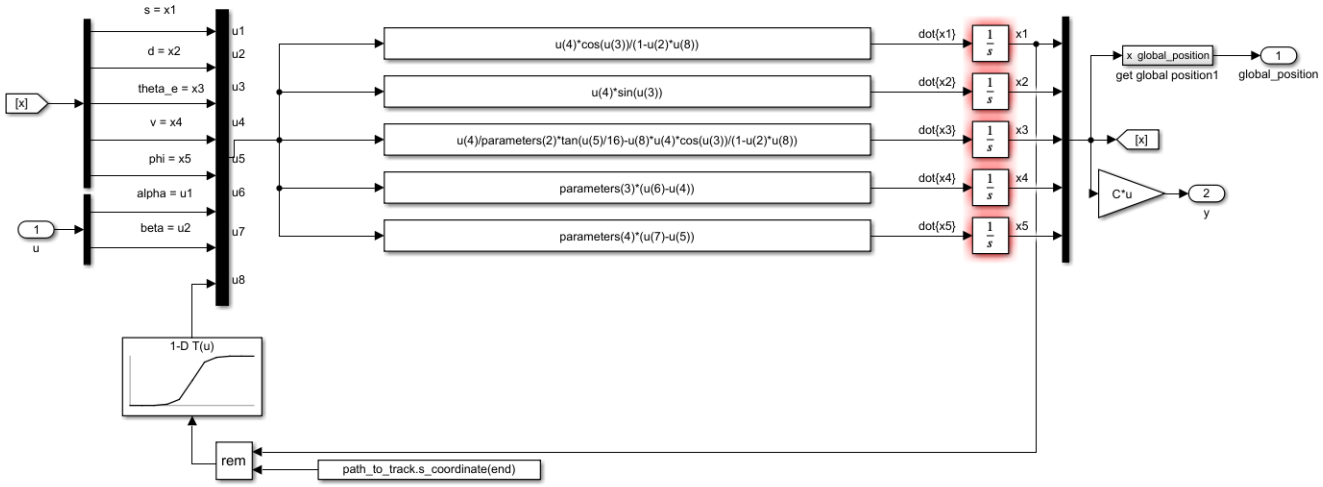

Figure 1: Implementation of the Non linear model-continuous

1.3 Provide a screenshot of the block diagram implemented in open loop experiments.slx.

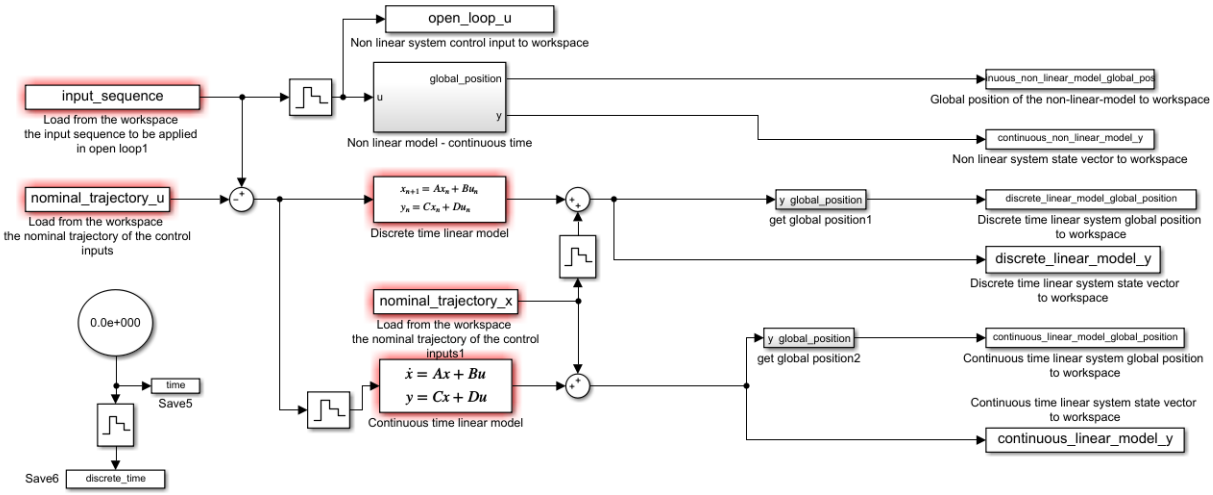

Figure 2: block diagram implemented in open loop experiments.slx

1.4 Report the simulation results, including the information concerning the initial state you used, the sequence of inputs you applied, and the simulated results.

Simulation 1 : Different simulations have been made. For the first one, the initial state is equal to the first point of the nominal trajectory : $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 5 \\ 6.4e-9 \end{bmatrix}$ and the inputs are constant and equal to my control input : $u = \begin{bmatrix} 0.5 \\ 6.4e-9 \end{bmatrix}$

Using those inputs and this initial state, the simulation results are :

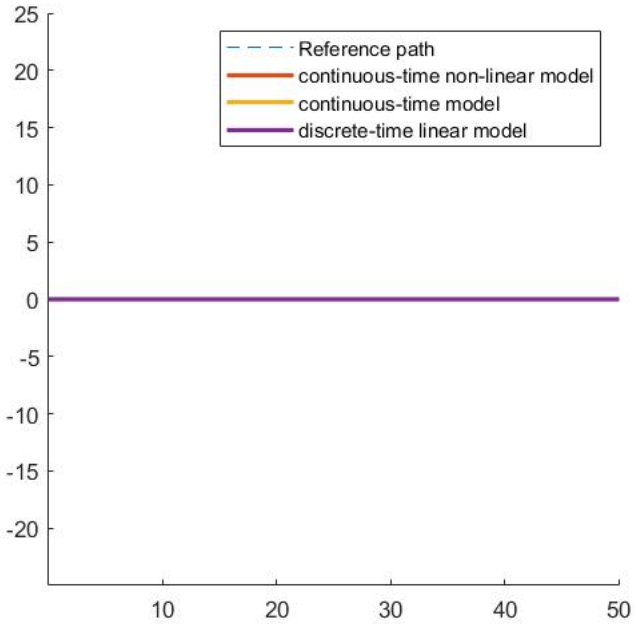

Figure 3: trajectory of the car using initial state and the sequence of input of simulation 1

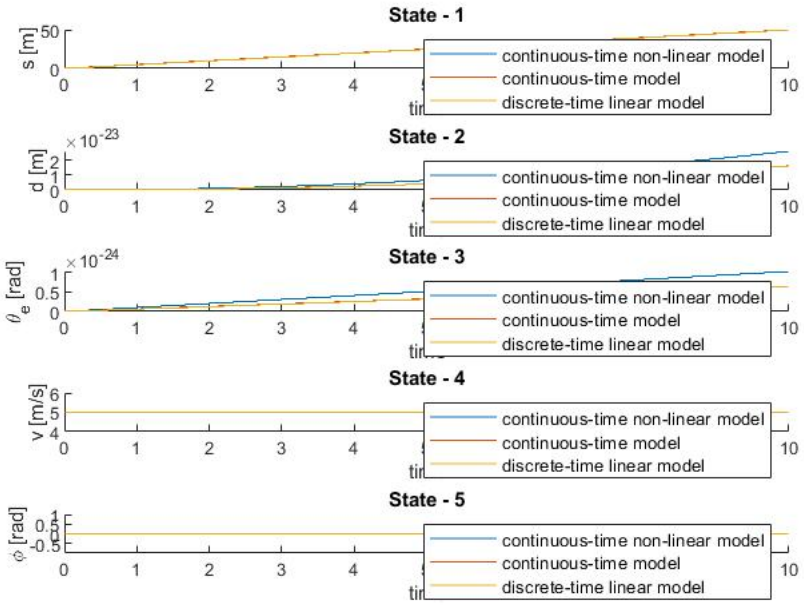

Figure 4: evolution of the state of the system using initial state and the sequence of input of simulation 1

The final value for the state that we obtain after a simulation of 10 seconds are :

$$x_{dis\_lin}(end) = \begin{bmatrix} 50.000 \\ 1.6146e-23 \\ 6.4623e-25 \\ 5.000 \\ 6.400e-9 \end{bmatrix} \quad x_{cont\_lin}(end) = \begin{bmatrix} 50.000 \\ 1.6153e-23 \\ 6.4623e-25 \\ 5.000 \\ 6.400e-9 \end{bmatrix} \quad x_{con\_nonlin}(end) = \begin{bmatrix} 50.000 \\ 2.5849e-23 \\ 1.034e-24 \\ 5.000 \\ 6.400e-9 \end{bmatrix}$$

Simulation 2 :

For the second simulation, we keep the input sequence equal to the control inputs : $u = \begin{bmatrix} 0.5 \\ 6.4e-9 \end{bmatrix}$ but we modify the initial state: $\begin{bmatrix} 0 \\ 0.1 \\ 0.05 \\ 0 \\ 0 \end{bmatrix}$

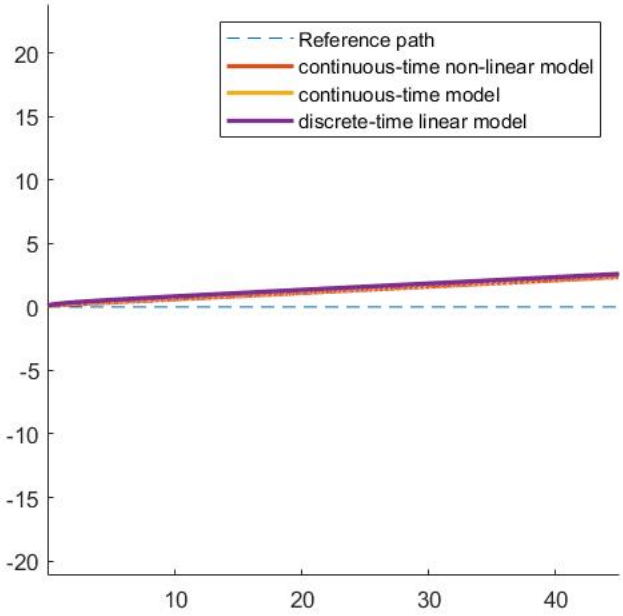

Figure 5: trajectory of the car using initial state and the sequence of input of simulation 2

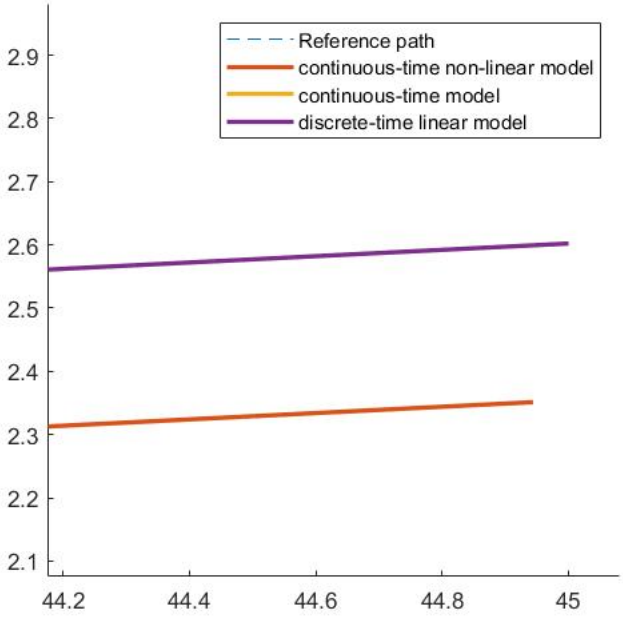

Figure 6: trajectory of the car using initial state and the sequence of input of simulation 2

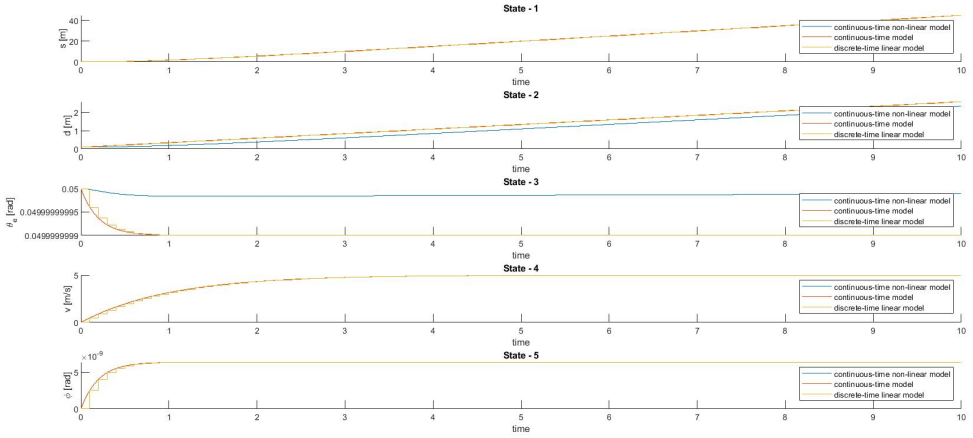

Figure 7: evolution of the state of the system using initial state and the sequence of input of simulation 2

The final value for the state that we obtain after a simulation of 10 seconds are :

$$x_{dis\_lin}(end) = \begin{bmatrix} 45.0002 \\ 2.6000 \\ 0.0500 \\ 4.9998 \\ 6.400e-9 \end{bmatrix} \quad x_{cont\_lin}(end) = \begin{bmatrix} 45.0002 \\ 2.6000 \\ 0.0500 \\ 4.9998 \\ 6.400e-9 \end{bmatrix} \quad x_{con\_nonlin}(end) = \begin{bmatrix} 44.9440 \\ 2.3491 \\ 0.0500 \\ 4.9998 \\ 6.400e-9 \end{bmatrix}$$

simulation 3 :

For the last simulation, we keep the initial state equal to the first value of the first point of the nominal trajectory : $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 5 \\ 6.4e-9 \end{bmatrix}$ and we made the sequence of input evolve over time :

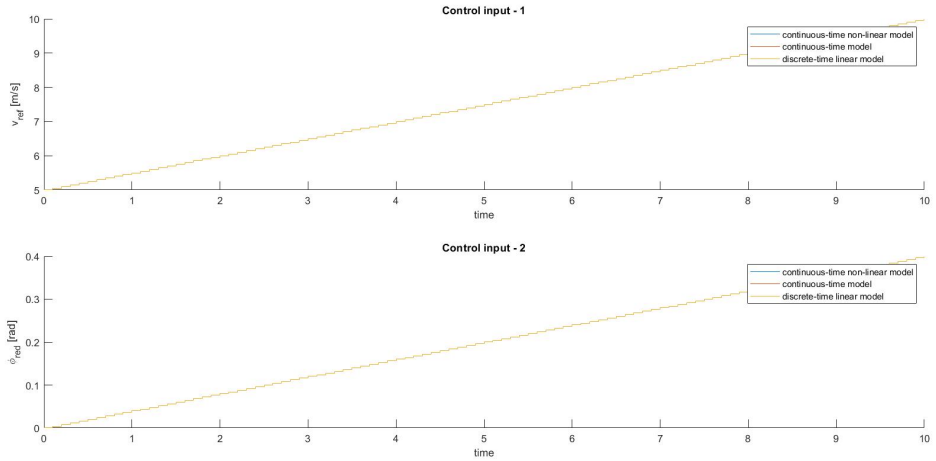

Figure 8: sequence of input used for simulation 3

Here are the results that we obtain after the simulation :

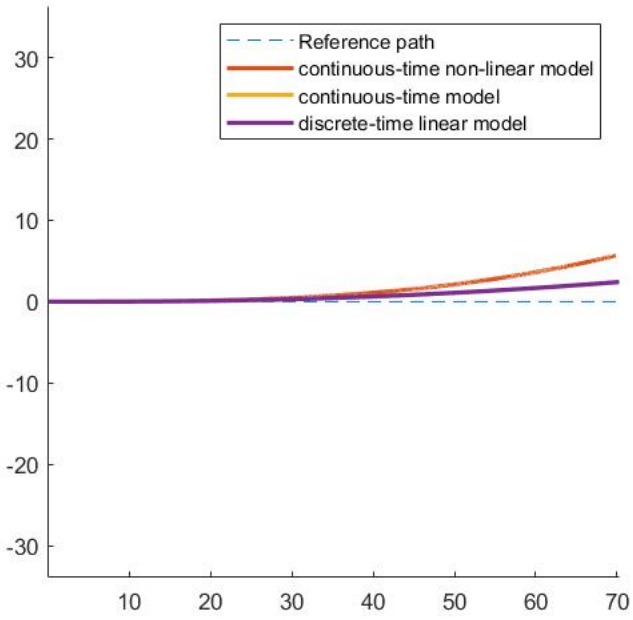

Figure 9: trajectory of the car using initial state and the sequence of input of simulation 3

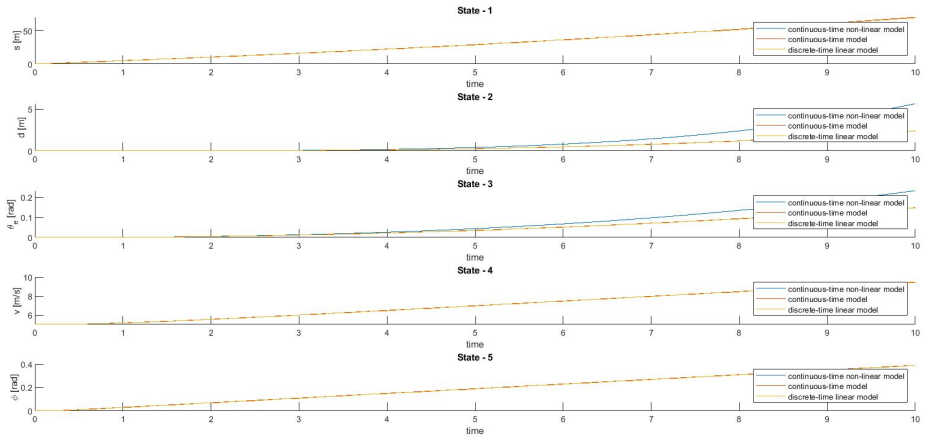

Figure 10: evolution of the state of the system using initial state and the sequence of input of simulation 3

The final value for the state that we obtain after a simulation of 10 seconds are :

$$x_{dis\_lin}(end) = \begin{bmatrix} 70.2779 \\ 2.4171 \\ 0.1486 \\ 9.4747 \\ 0.3898 \end{bmatrix} \quad x_{cont\_lin}(end) = \begin{bmatrix} 70.2779 \\ 2.4171 \\ 0.1486 \\ 9.4747 \\ 0.3898 \end{bmatrix} \quad x_{con\_nonlin}(end) = \begin{bmatrix} 69.8734 \\ 5.6665 \\ 0.2336 \\ 9.4747 \\ 0.3898 \end{bmatrix}$$

1.5 Does the linear model resemble sufficiently well the behavior obtained from the non-liner model? Why?

Looking at simulation 1, we observe that the linear and non linear model have the same behaviour. Since the initial state is equal to the first point of the nominal trajectory and that our sequence of inputs is equal to our control inputs, our states at time t will always be equal to the nominal trajectory at time t. In this case, we have $\tilde{x}(t) = 0$ and we will have no error due to linearization.
Looking at simulation 2 and 3 we observe that when our initial states is different from the first point of the nominal trajectory or when our sequence of inputs is different from the control input, we obtain different behaviours for our linear and non linear model. Moreover, we observe that the differences in the states for the linear and non linear case increase with time. This is due to the fact that in the linear case, the states of the car will not be equal to the nominal trajectory and we will get some error in our linearization. Those errors will accumulate and will increase with time.

To conclude, the linear model resemble well the behaviour obtained from the non linear model when the car stay close to the nominal trajectory.
1.6 Do the results obtained using the discrete-time linear model resemble those obtained by the continuous-time linear model? What is then interesting about using the discrete-time linear model instead of the continuous-time linear model to design control strategies?

Looking at the three simulations made above we observe that the order of the difference between the discrete time linear and the continuous time linear model is equal to :

simulation	order of the error
1	0
2	e-12
3	e-13

Table 1: Error between the discrete-time linear model and the continuous-time linear model

We can conclude that the results obtained using the discrete-time linear model resemble those obtained by the continuous-time linear model.

Using discrete time linear model instead of continuous time linear model can be interesting because they give directly as output the state and have a smaller computation cost.

Remark: The discrete-time linear model resemble those obtained by the continuous-time linear model because the time step is small enough. If we take a larger value for the time step, we will get larger error between the discrete time linear model and the continuous-time linear model.

4 Session 2 : Control and observation

2.1 Report the value of the LQR gain for the proposed Q1 and Q2, as well as the evolution of the singular values of S obtained while solving the Riccati equation.

The value of the LQR Gain is equal to

$$K = \begin{bmatrix} 0.0032 & 0.0000 & 0.0000 & 0.2259 & 0.0000 \\ 0.0000 & 199.0563 & 722.291 & 0.000 & 19.4736 \end{bmatrix}$$

and here is the evolution of the singular values of S obtained while solving the Riccati equation :

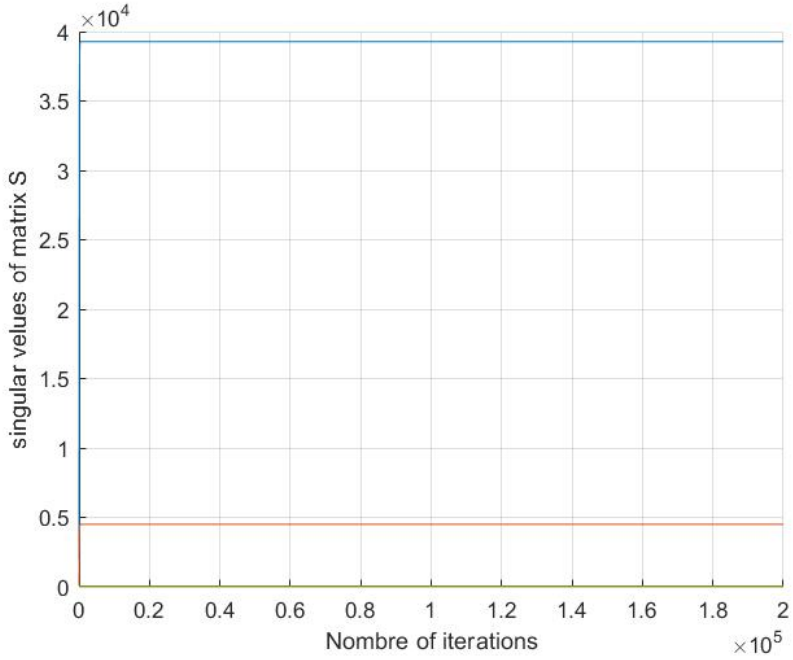

Figure 11: evolution of the eigenvalues of the S matrix

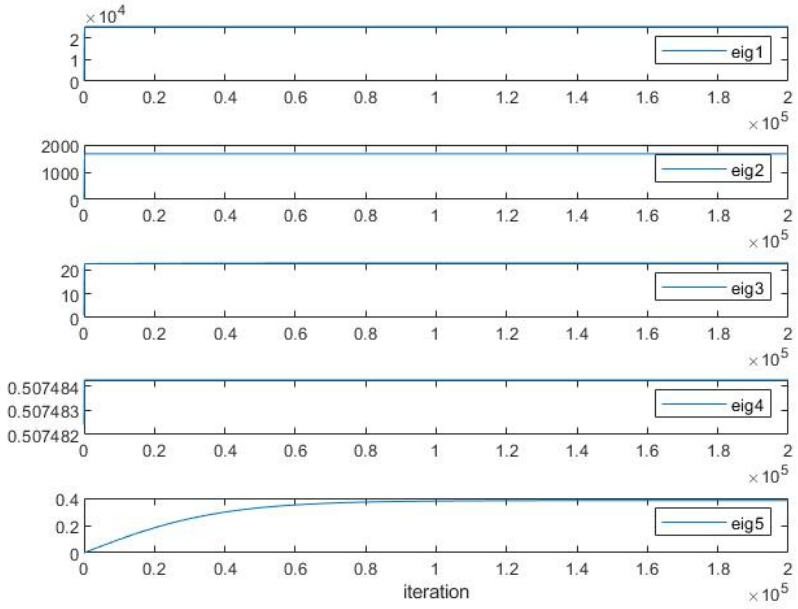

Figure 12: evolution of the eigenvalues of the S matrix

2.2 Report the explicit value of the observation close loop poles, and the observer gain resulting from it.

For the moment, we set the poles of the observation loop as the 99.9% of the poles of the closed control.

We find :

$$pole_1 = 0.9900 + 0.0000i$$
$$pole_2 = 0.9779 + 0.0000i$$
$$pole_3 = 0.0153 + 0.0000i$$
$$pole_4 = 0.9762 + 0.0136i$$
$$pole_5 = 0.9762 - 0.0136i$$

Using the matlab function place we find :

$$\text{L}= \begin{bmatrix} 0.98451 & 2.7146e-12 & 0.0099502 & 3.2149e-17 \\ -1.6076e-12 & 0.029931 & 0 & 0.00001921 \\ 2.4774e-13 & 0.0082671 & 0 & 0.00076204 \\ 0 & 0 & 0.0032104 & 0 \\ 0 & 0 & 0 & -0.047745 \end{bmatrix}$$

2.3 Provide a screen shot of your final Simulink scheme and the submodules you had to complete.

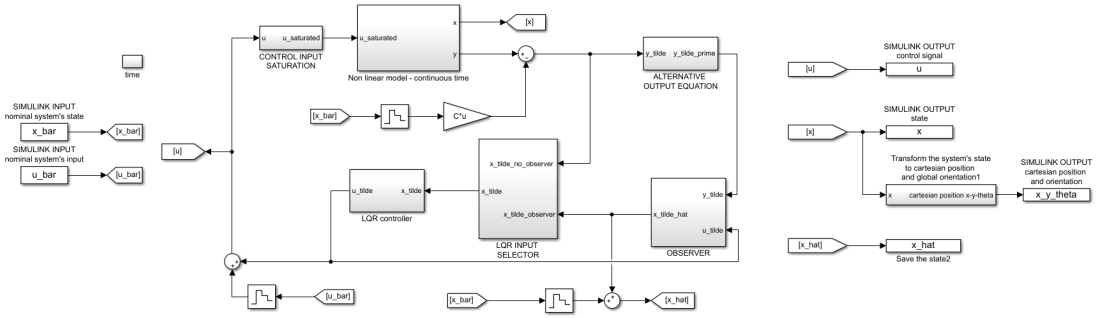

Figure 13: LQR˙observer.slx

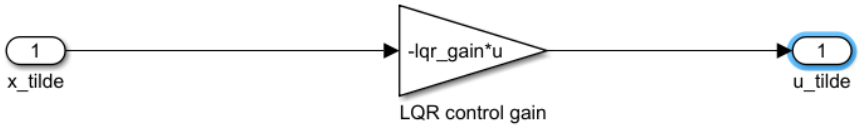

Figure 14: LQR controller

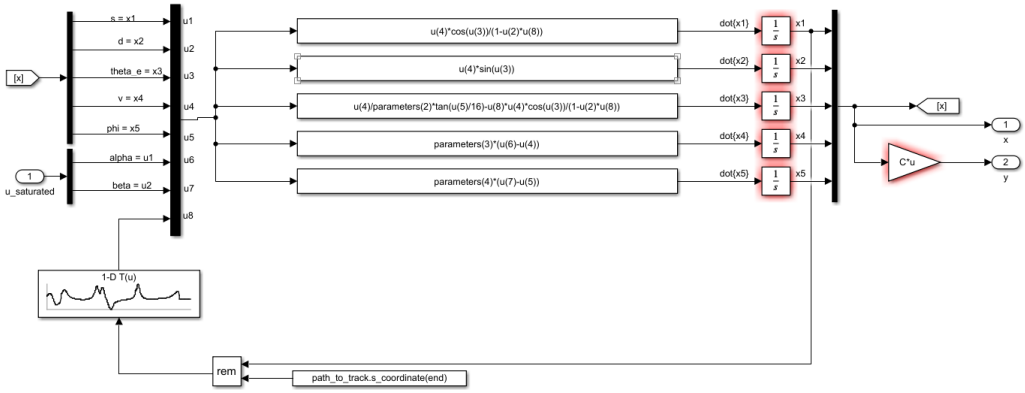

Figure 15: Non linear model- continuous time

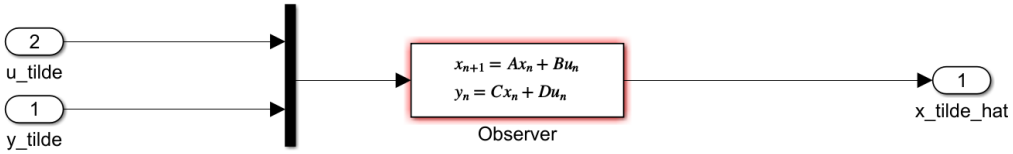

Figure 16: Observer

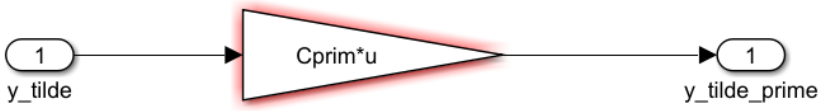

Figure 17: Alternative output equation

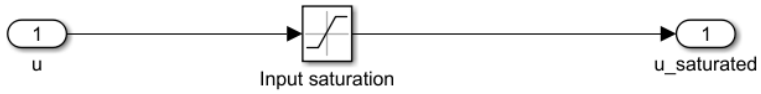

Figure 18: Input saturation

2.4 Report the simulation results for the proposed values

simulation 1 : The first simulation was made using the path1 as the reference path. The initial value for x0obs is equal to $x0obs = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$. The initial state is equal to $x0 = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}$

Here are the results obtain with this initial configuration :

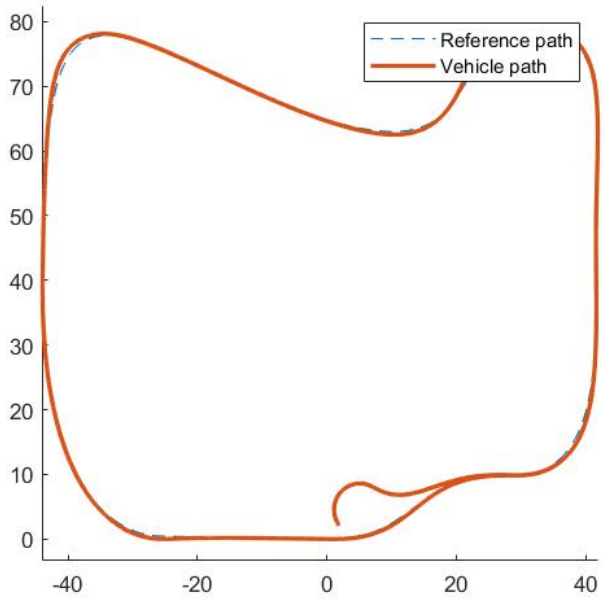

Figure 19: vehicule path with fully observable state

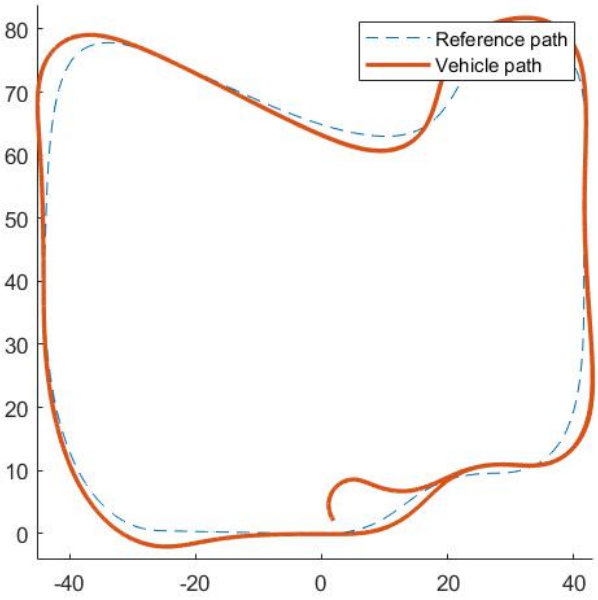

Figure 20: vehicule path with LQR + observer

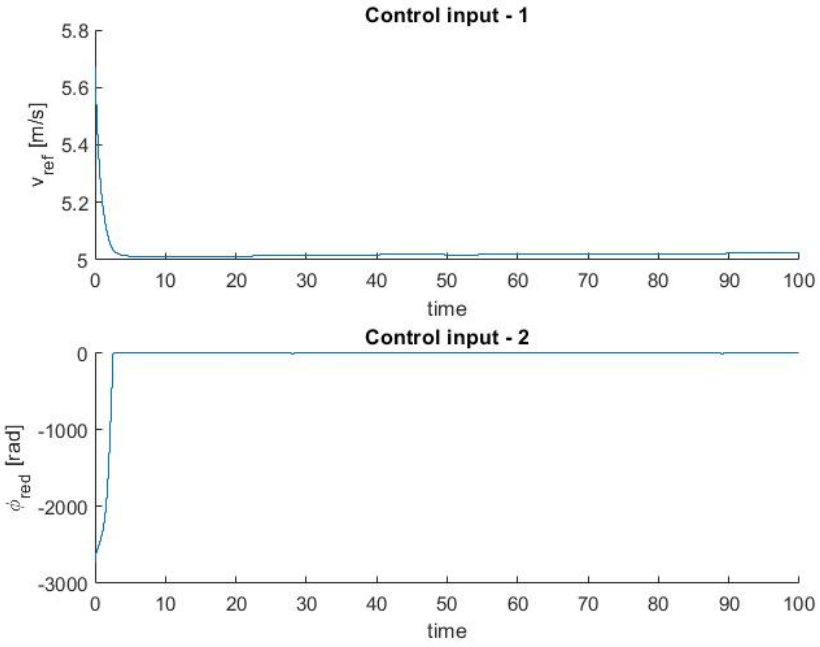

Figure 21: Input for fully measurable state

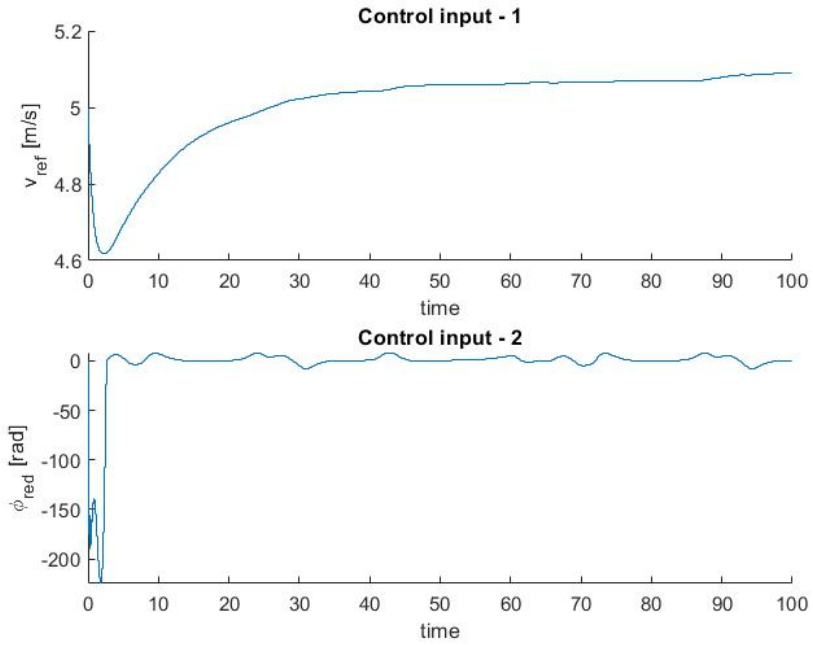

Figure 22: Input for LQR + observer

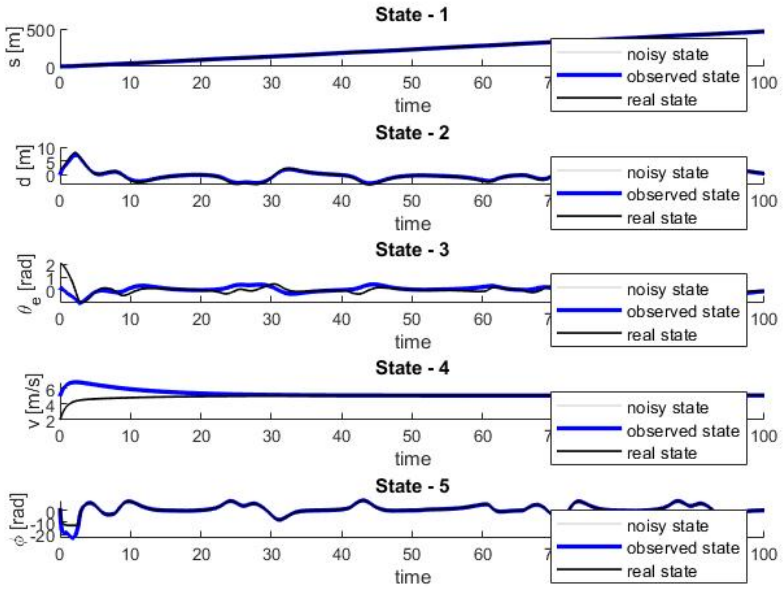

Figure 23: State of the vehicule

Other simulations :
For the 3 others simulation, we use the same x0obs, x0 :

$$x0obs = \begin{bmatrix} 1 \\ 0.5 \\ 0.8 \\ 10 \\ 0.9 \end{bmatrix}, \; x0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 5 \\ 6.4e-9 \end{bmatrix}$$

Using the circle path we find :

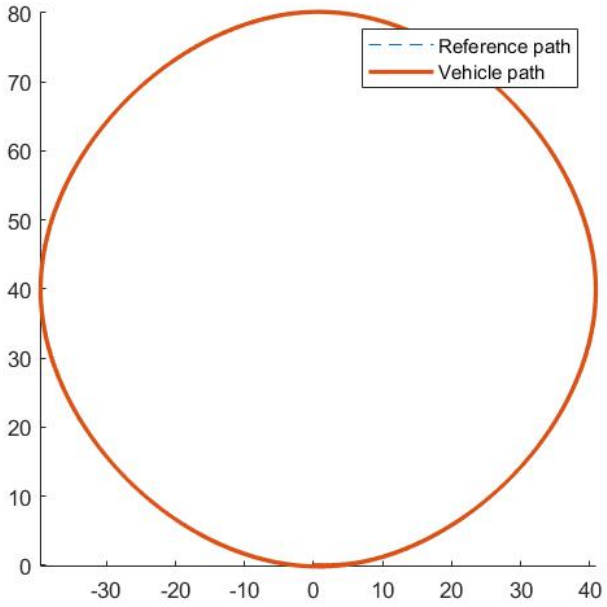

Figure 24: vehicule path with fully observable state

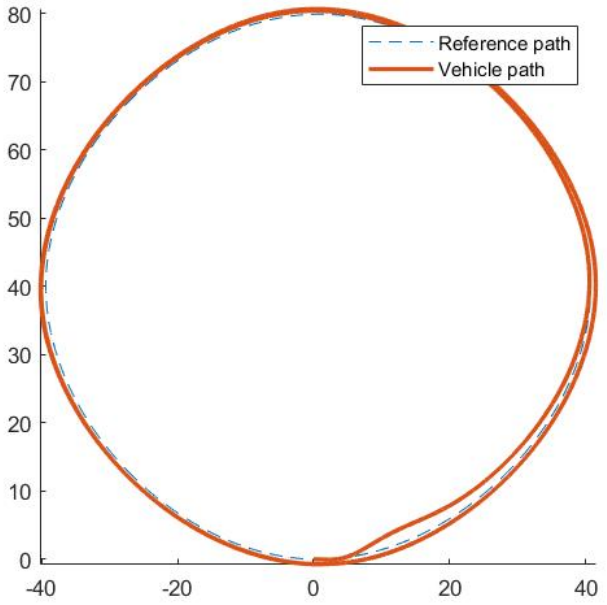

Figure 25: vehicule path with LQR + observer

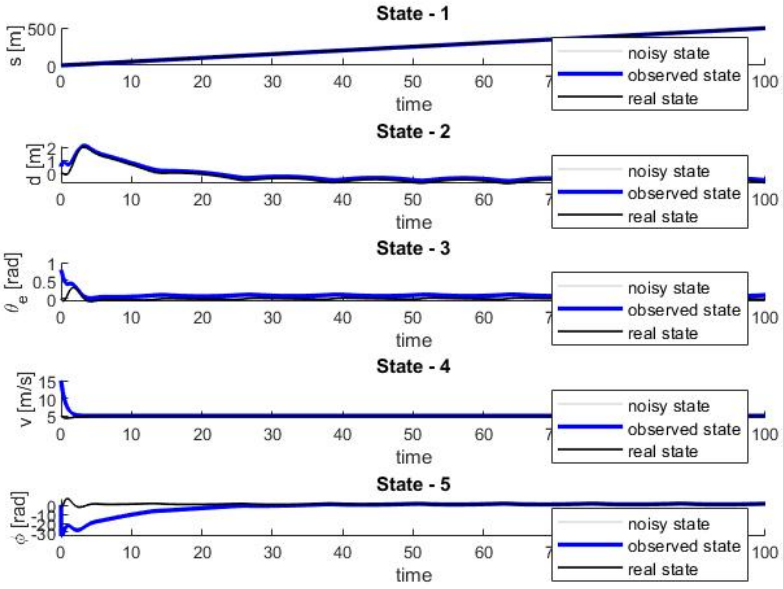

Figure 26: State of the vehicule

Using the path 2 on matlab we find :

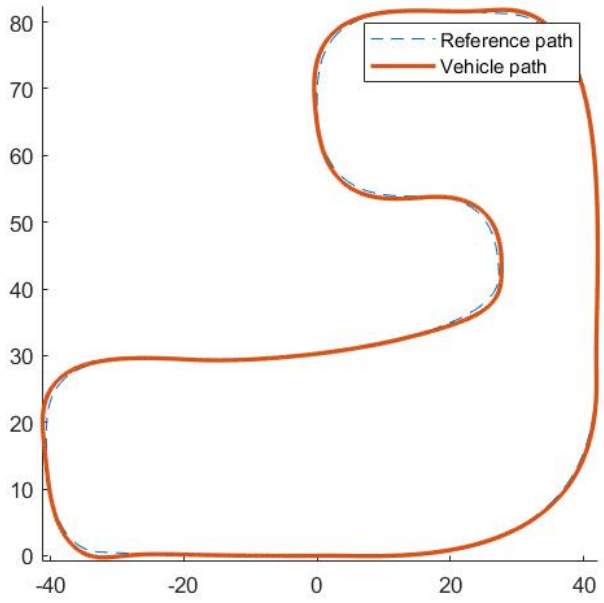

Figure 27: vehicule path with fully observable state

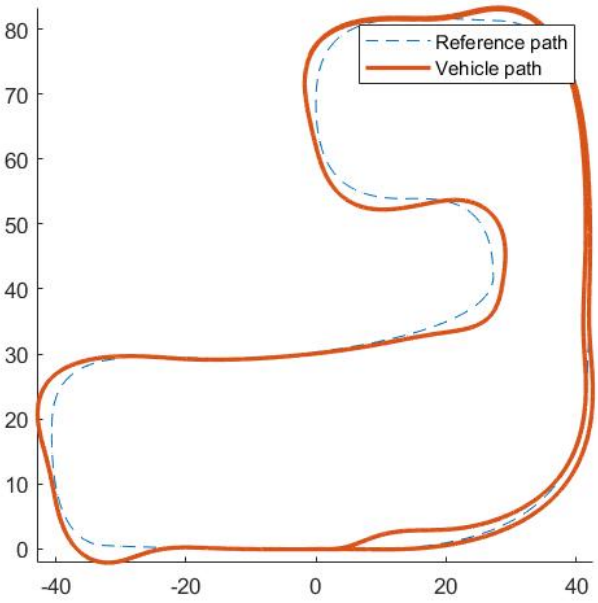

Figure 28: vehicule path with LQR + observer

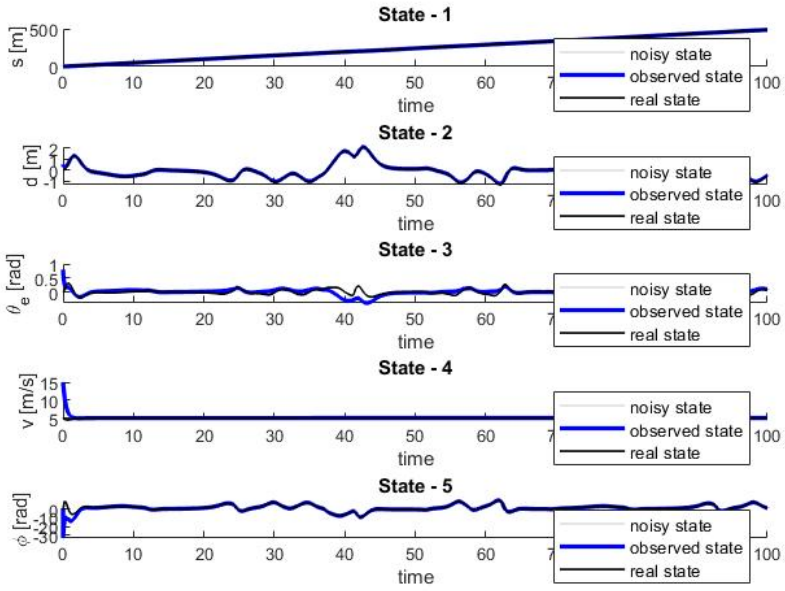

Figure 29: State of the vehicule

Using path 3 on matlab we find :

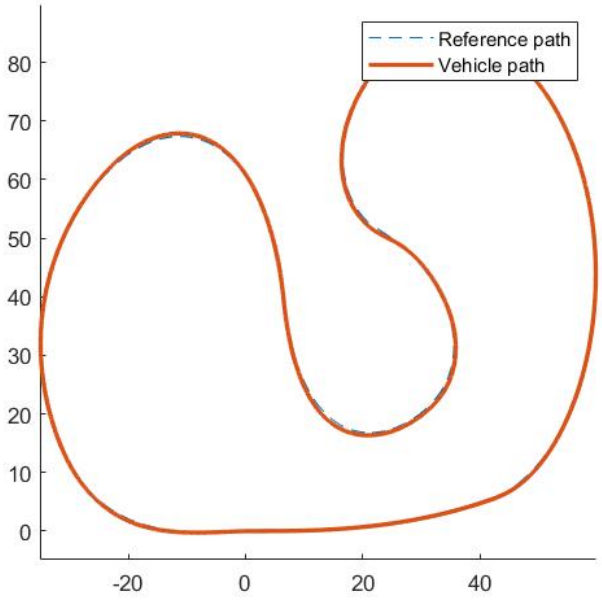

Figure 30: vehicule path with fully observable state

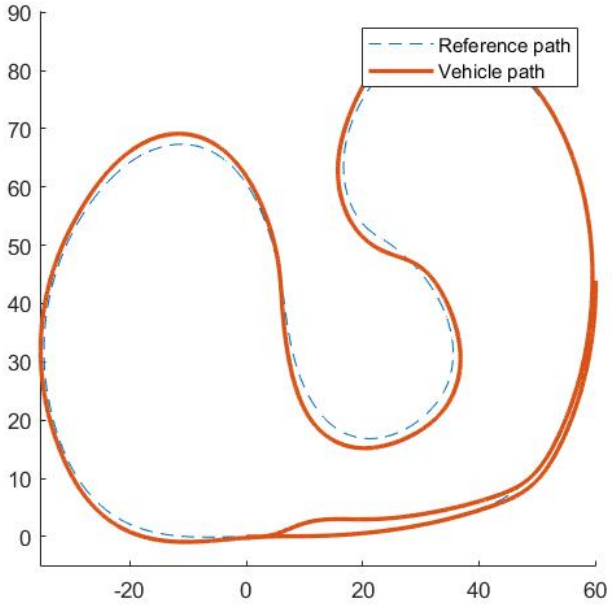

Figure 31: vehicule path with LQR + observer

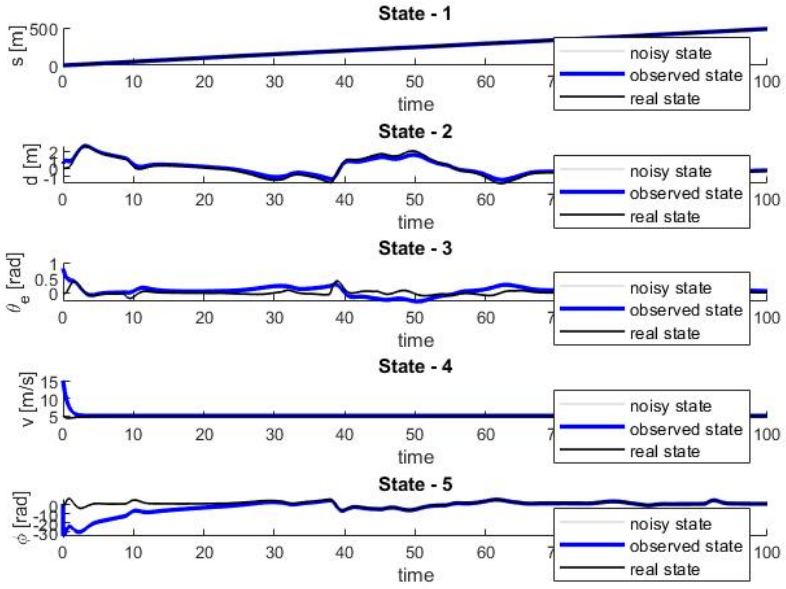

Figure 32: State of the vehicule

Looking at those four simulations, we observe that our model is working even if we have wrong inital states and observability initial states different from the system's real initial state.

Looking at the case where one state is not observable and where an observer is added, we observe that our car is getting away from the reference path when the car is turning. This is due to the fact that the curvature of the path change and the observer need some time to correct the error and get the correct value for x_3.

2.5 The proposed value of Q1 intends to assign very little importance to the error deriving from x1. Could you explain why doing this might make sense?
x_1 gives the distance traveled by the car and has no influence on the other state. Moreover, correction of the distance travelled is not the priority as it has a small influence comparing to the other parameters on the tracking of the referene path.
It is more important to get a correct values for the state x_2, x_3, x_4 and x_5 since they will allow the car to be in the good direction, at the good position and at the correct speed and therefore follow the reference path.

6. Propose a different pair Q1 and Q2 such that heading deviation error is highly penalized compared to the other states and report simulation results where the impact of doing so can be clearly observed.
The gain matrix K is calculated so that the quadratic cost $J(u) = x^T Q_1 x + u^T Q_2 u$ is minimized. If we want to penalized the heading deviation error more that the other states we have to increase the value $Q_1(3,3)$.

A different pair Q_1 and Q_2 could be : Q_1=diag([0.0001 50 1000 0.5 0.5]) and Q_2=diag([1 0.00002]).

With those two matrices, if we run the simulation using the path 2 and by using $x0obs = \begin{bmatrix} 1 \\ 0.5 \\ 0.8 \\ 10 \\ 0.9 \end{bmatrix}$,

$$x0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 5 \\ 6.4e-9 \end{bmatrix}$$ we find :

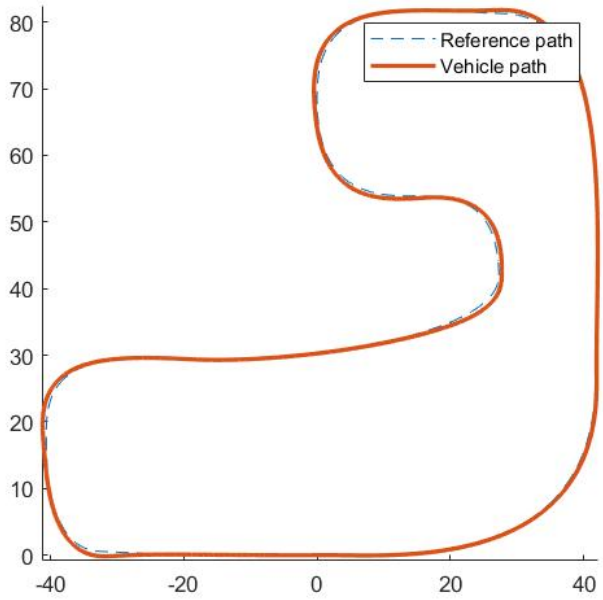

Figure 33: trajectory for path2 with new matrix Q1 and with fully observable state

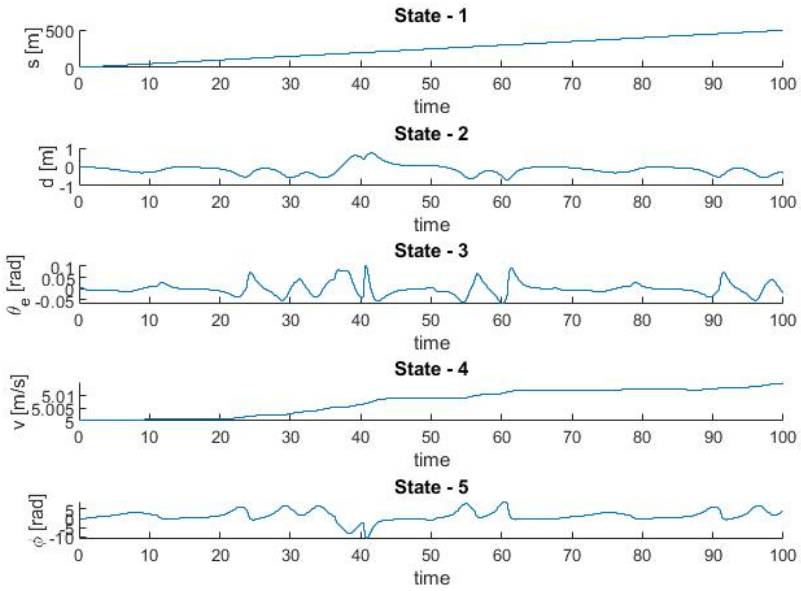

Figure 34: state for path 2 with new matrix Q1 and with fully observable state

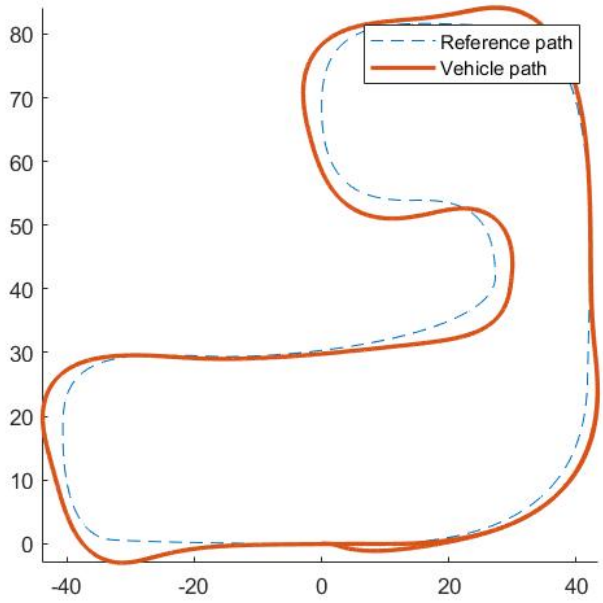

Figure 35: trajectory for path2 with new matrix Q1 LQR + observer

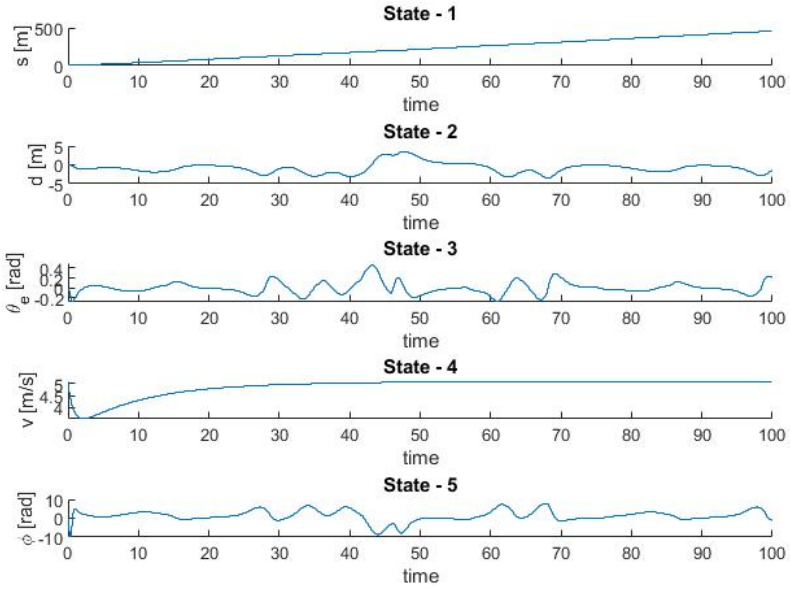

Figure 36: state for path2 with new matrix Q1 LQR + observer

Looking at figure 34 we can see that when the car needs to turn it gets away from the reference path. This is due to the fact that the state θ_e is computed thanks to an observer and therefore there will be a small error in the estimation of θ_e when the curvature of the path changes. Since we changed the matrix Q1 so that an heading deviation error is highly penalized our car will try to correct θ_e in priority compared to the other state even if there is some error in the estimation of θ_e. We can confirm this by looking at figure 35. We observe that θ_e is always small but we have large values for the lateral error.

2.6. Is the proposed placement for the observation close loop poles appropriate? why? If not, propose a new set of poles to improve the observation performance.

Using the observer, the error in the state can be given as :

$$\delta(k+1) = (\Phi - Lc^T)\delta(k)$$

To make this error goes as fast as possible to zero, the eigenvalues of $\Phi - Lc^T$ must be placed at 0+i0. Nevertheless putting those eigenvalues close to 0 make the system sensitive to noise (In the simulation we have no noise in the measurement but we will have noise in the real world). We can therefore reduce the value of the pole and set the poles of the observation loop as 65.5% (instead of 99.9%) of the poles of the closed control loop.
Here is the path obtain with the same parameters as in question 2.6 and with Q1=$diag$([0.00015010000.50.5]):

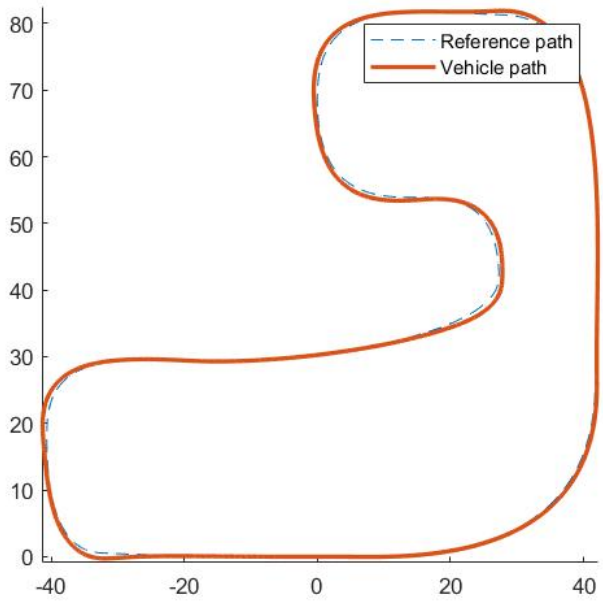

Figure 37: trajectory for path2 with new placement for the observation close loop poles

We observe that we obtain a better path than in figure 34 due to the fact that errors in the state are corrected faster.

5 Conclusion

For this case study, in a first time, we built different open loop models for the simulation. We observed that we get similar result between non linear and linear model when the initial state and the input are equal to the nominal trajectory and the nominal input.
In a second time we designed an LQR controller and an observer and we apply them to our path tracking problem. We realized that parameters in this design are tunable and modifying their values improve or decrease the performance of our model.