MULTIVARIABLE CONTROL AND COORDINATION SYSTEMS (EE-477)

Case study:

Path tracking for an automated vehicle.

`http://react.epfl.ch`

Denis Gillet, Ezequiel Debada
Lausanne, EPFL, 2018

Exercise 2

Control and observation

In this session we will tackle the design of a LQR controller and an observer to address the path tracking problem. The main idea is using the discrete-time linear model of the system to design a controller that would then be implemented to control the nonlinear system. We will explore the usefulness of observers by seeing the effect of measurement noise and non-measurable states on the controller performance, and how the observer helps in those circumstances.

Two Simulink models are going to be implemented in this exercise. In both of them you will implement a LQR closed-control loop, which will be complemented with a state observer in the second one.

2.1 Provided files

- `LQR_closed_Loop.slx` and `LQR_observer.slx`. Simulink files with some of the blocks needed to simulate the system. `LQR_closed_Loop.slx` only implements the LQR-control loop while `LQR_observer.slx` will additionally include the state observer.
- `ex2.m`. Class whose methods have to be completed. A description of every function to be completed is included within the file.
- `exercise2_LQRandObserver.m`. Matlab script that sets up and runs the required simulations. Modify its content only if you want to run other experiments in addition to the ones proposed.
- `utilities.m`. Class that gathers auxiliary functions that will be used in the scripts provided during the case study sessions. There is no need to review its content unless you want to make use of some of the functions included therein.
- `circle, path_1, path_2, path_3` mat files containing different paths to track that can be used in the experiments.

2.2 Exercises

LQR Control design and implementation

- Copy the content of functions `getSystemParameters`, `getLinealModelArrays`, `getDiscreteLinearModel`, and `getWorkingTrajectory` from the previous exercise.
- Complete the method `getCostFunctArrays` which returns the matrices Q_1 and Q_2 defining the cost function of the control problem. You can freely give values to those matrices, however you must neglect the error of x_1. Could you explain why?
- Complete the method `calculateLQRGain` which implements two different methods to calculate the LQR control gain.
- Complete the method `select_reference_path` which should return the name of the *.mat* file containing the information of the path that is wanted to be tracked.

- Complete the method `getInitialState` considering the initial state of the system being $x(0) = [0, 0, 0, 0, 0]$.
- Complete the Simulink diagram `LQR_closed_Loop.slx` implementing the LQR control loop.
- Simulate and observe the performance of the control approach. Run additional experiments to observe how the values of Q_1 and Q_2 impact the control performance.

LQR assessment with imperfect information

- Complete the method `getNoiseModule` which returns a structure containing the characteristics of the measurement noise. You can start by implementing a noise of zero mean and standard deviation $(1, 1, 0.1745, 2, 0)$ for states 1 to 5.
- Simulate and observe how the performance changes when the LQR deals with noisy signals.
- Complete the method `getCprime` which returns C', a version of the C array that allows to emulate a situation where a certain state is non-measurable. This matrix must have the same dimensions than C but with the rows affecting the targeted state equal to zero. The objective is making the value of the output that arrives to the LQR be $\tilde{y}' = C'\tilde{y} = C'\tilde{x}$. In particular, we will consider that we do not have available information regarding x_3.
- Make sure that in the Simulink diagram, the state information that arrives to the LQR controller is $\tilde{x}' = C'\tilde{x}$.
- Simulate and observe the effect of using the LQR control strategy when one state is missing.

State observer design and implementation

- Complete the method `getCArrayConsideringMeasurableStates` which returns a third version of the C array (here referred to as C_{mes}) that allows to reduce the state vector by neglecting the states that are considered unmeasurable. The main difference between this and C' array defined in the previous method is indeed the dimension of the resulting output vector. Specifically we will consider that x_3 cannot be measured. Keep also in mind that the state vector that is expected to be received by the observer should be $\tilde{x}' = C_{mes}C'\tilde{x}$.
- Complete the method `checkObservability` which returns the number of states that are not observable as a result of the state considered immeasurable.
- Complete the method `getObserverGain` calculate the observer gain and implement the observer in Simulink.
- Complete the Simulink diagram `LQR_observer.slx` which should implement the LQR-control loop with a state observer. Keep in mind that (i) the LQR controller is fed by the state observer, (ii) that the observer is designed w.r.t. $\tilde{x}$ and $\tilde{u}$, (iii) and that the state vector that arrives to the observer should be $\tilde{x}' = C_{mes}C'\tilde{x}$.
- Simulate the observer and compare the performance with the previous exercise in which the observer was not implemented yet the same state was considered to be immeasurable.
- Repeat the experiment by making the dynamic of the observer (which depends on the position of the close loop poles) faster/slower and observe the impact on the state prediction and control.