Assignment 2: Introduction to Controller Design and Stateflow

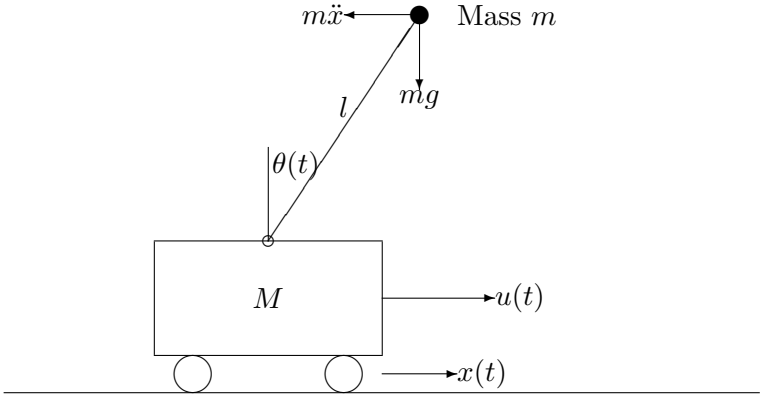
1DT059: Model-based Design of Embedded Software
2022

October 7, 2022

In this assignment you produce Simulink models and text. Your solutions to each exercise should be implemented in a .slx-file and/or .pdf-file that is named after the exercise (the solution to assignment 2, problem 2 should be in file `a2e2.slx` and/or `a2e2.pdf`). Include your name at the top of all submitted files.

Assignment 2 is to be solved by students individually.

Problem 1: Inverted Pendulum (25p)
Assume a system consisting of an inverted pendulum mounted on a cart, that can be moved by applying a horizontal force. I.e., the system looks like the one in the below figure.



The system consists of a cart, with mass M, to which is attached a rod (or pendulum) of length l. At the end of the rod is a smaller mass m. The mass of the rod can be neglected. The cart can move horizontally. We let $x(t)$ be the position of the cart at time t, with the positive x-direction going to the right. The rod can rotate in the xy-plane around an axis located where the rod is connected to the cart. The input to the system is the horizontal force

$u(t)$ that is applied to the cart. The observed variables of the system are the horizontal position $x(t)$ of the cart and the angle $\theta(t)$ of the rod that is mounted on the cart. We can take $\theta(t) = 0$ when the rod is upright, and let θ be positive when the rod tilts to the right, as in the above figure. The primary objective of control is to keep the pendulum upright (i.e., making $\theta(t) = 0$) by controlling $u(t)$. A secondary objective is to also control the position $x(t)$ of the cart.

We begin by modeling the system. The external force in the horizontal direction is u. By Newton's second law, this corresponds to a horizontal acceleration $\ddot{x}$ of the cart and mass (we approximate the total mass of the cart and small mass by M) and also an additional horizontal acceleration approximately equal to $l\ddot{\theta}$ of the mass m due to swinging of the rod. This gives the equation

$$M\ddot{x} + ml\ddot{\theta} = u$$

We also set up equations modeling the torques around the pivot point, where the rod meets the cart. Here the gravity on the mass gives a torque $mlg\theta$ (approximating $\sin(\theta)$ to be θ), and the acceleration of the cart gives the torque $ml\ddot{x}$. Since the angular inertia of rod and mass is ml^2, this gives the equation

$$ml^2\ddot{\theta} = mlg\theta - ml\ddot{x}$$

Solving for $\ddot{\theta}$ in the first equation, and for $\ddot{x}$ in the second one, we obtain

$$\begin{array}{rcl} (M-m)\ddot{x} + mg\theta & = & u \\ (ml^2 - \frac{m^2l^2}{M})\ddot{\theta} - mlg\theta & = & -\frac{ml}{M}u \end{array}$$

or

$$\begin{array}{rcl} (M-m)\ddot{x} + mg\theta & = & u \\ l(M-m)\ddot{\theta} - gM\theta & = & -u \end{array}$$

As actual values, we can take M is 1kg, m is 0.1kg, and l is 0.6m.

Exercise 1: We will consider that either the angular position θ or the angular velocity $\dot{\theta}$ can be measured by the controller.

a) From the above equations, produce a continuous-time model in state-space form, and use it to make a Simulink model of this system (using a state-space block).

b) Assume that only the angular position θ of the pendulum can be observed (i.e., measured). Try to design (and tune) a P, PI, or PID controller that keeps the pendulum upright. Is it possible to do this using a PID controller? Using a PI controller? Check that it can put the pendulum upright even if there are disturbances. E.g., check that the pendulum will become upright if started in a tilting position (or add a disturbance signal to u).

c) Same question as in b), but now for the case that only the angular velocity $\dot{\theta}$ is observable. Is there a difference in response compared to b)? Why?

d) After having designed the controller to make θ be 0, see what happens to the position x when you started in a tilted initial position. Will x stay in its initial position?

Exercise 2: We continue using the same system as in Exercise 1. This time, however, your task is to control both the angle θ and the position x.

a) Assume that all state variables (i.e., $x\ \dot{x}\ \theta\ \dot{\theta}$) can be observed. Design a full state-space feedback controller using pole placement. Put poles so that the behavior of u and θ are reasonably smooth. (in practice there will be some limit on the magnitude of the u signal).

b) Solve the same problem as in a), but this time using an LQR controller. This controller minimizes, the error

$$\int_0^\infty (\dot{x}^2 + \lambda_1\theta^2 + \lambda_2 u^2)dt$$

over the course of a simulation. Choose suitable cost coefficients for generating the LQR controller. After this, compare the cost that is minimized by the LQR design to the same cost for the controller that you generated in a).

c) Assume next that only some state variable(s) are observable. To find out what you need to observe, find out for which combinations of state variables (i.e., $x\ \dot{x}\ \theta\ \dot{\theta}$) the system is observable.

Problem 2: Microwave Oven (25)
Consider a simple version of a controller for a microwave oven. The microwave is used to cook food items for specified periods of time at specified levels of power. A user can enter desired cooking time and cooking power, and start or stop the oven. To make the problem more specific.

- The oven allows the user to select cooking time. For simplicity, the cooking time can be given in seconds (up to, say, 1000s). The setting of cooking time is done by two buttons: one for increasing the time (in suitable steps) and one for decreasing the time. The remaining cooking time is displayed on the oven. The initial (default) value is 0.
- The oven also allows the user to select cooking power, as any integer between 0 and 800. The current cooking power is displayed on the oven. The initial (default) value is 0.

- The oven has some buttons:

  Start for starting or resuming cooking. During cooking, the display for remaining time is decreased with the passage of time.

  Stop for stopping the cooking. If the oven is cooking, this suspends cooking and stops the decrease of time on the time display. If the oven is not cooking, the **Stop** button acts a reset, which resets the remaining time to 0.

- The oven has a door which can be opened or closed by the user. Opening the door suspends cooking and stops the decrease of time on the time display. Closing the door does not restart the cooking: for that the **Start** button must also be pressed.

- When the time display reaches 0, the oven sounds an alarm, and stops the cooking.

- During cooking, the user can freely change the remaining time and cooking power. If the user changes remaining time to 0, the alarm sounds, and cooking stops (if it was active).

- If the user has been inactive for more than 300 seconds while the oven has not been cooking (e.g., the user has set the cooking time, but thereafter done nothing), the oven resets to its initial values.

Make a well structured Stateflow model of a microwave oven as above. Realize the interface by suitable interface blocks, that will allow you to interact with the model during simulation.

Problem 3: Elevator Controller (50p)

In this problem, you will construct a Stateflow model of a controller for a simple elevator. You will later (after some modification) be able to use this model in the project on implementing an elevator controller. The elevator serves four floors (numbered 1, 2, 3, 4). There are 4 buttons, one for each floor, that can be pressed by users (for simplicity, we do not distinguish between buttons at floors and buttons inside the elevator, there is just *one* button for each floor). Whenever a button for floor n has been pressed, a light connected to that button is lit, and the elevator eventually (possibly after passing and stopping at some other floors) arrives to floor n. When this happens, the button is "unpressed" (e.g., its light goes off) and the doors open. The state of doors should be modeled by another light, which goes on when doors are open, and which goes off when they are closed. The elevator then stays at this floor until the control unit orders it to move to some other floor. Once the doors are open, they remain open for at least 5 seconds. The elevator must close the doors before leaving a floor.

The control unit controls the elevator itself via an output signal *speed*, which is the speed of the elevator cart. For simplicity, we can use the unit

floors/*s*, where 0 speed means "stopped", positive speed means "going up", and negative speed means "going down". The maximum absolute speed of the elevator is 0.2*floors*/*s* (i.e., it takes at least 5 seconds to go from one floor to the next). The elevator informs the control unit about its position via the signal *position* (an input to the controller), whose value is a real number between 1 and 4 (i.e., the unit is *floors*).

Apart from the input signal *position*, the controller has input events generated by pushing buttons. These are the above mentioned buttons for each floor (*floorn* for $n = 1, 2, 3, 4$). There is also an emergency button. Whenever the "emergency" button is pressed, the elevator is stopped and the emergency lamp is lit. The lamp is switched off and the elevator resumes to normal when the "start" button is pressed again.

Should we have both a "stop" and an "emergency" button?

Output signals from the controller are (in addition to *speed*), one lamp (either on or off) for each floor button, as described above, one door lamp, indicating that doors are open, and an emergency lamp, indicating that the emergency button has been pressed.

a) (35p) Make a Stateflow model of the controller, which interacts with buttons, lamps, and a simple Simulink model of the elevator (with input *speed* and output *position*. Input from buttons should be in the from of triggering events that can enable transitions in the Statelof model. The requirements include typical "elevator-behavior" requirements, including that: whenever the button for floor n has been pressed, the elevator must eventually arrive to floor n within some reasonable time. When this happens, the "door" lamp is lighted, signalling that the doors are open. Once the doors are open, they remain open for at least 5 seconds. The doors must close (i.e., the lamp go off) before leaving any floor. Whenever the "emergency" button is pressed, the elevator is stopped and the emergency lamp is lit. The lamp is switched off and the elevator resumes to normal when the "start" button is pressed again.

   You should produce a well structured controller model, which is decomposed in a logical way following the functionality of the controller. In order that the Statechart can take input from both triggering events and from data signals, you should set its sample time to "inherited". One of the inputs to the trigger port should be a "clock", which triggers activation at frequent regular intervals. One subproblem is to decide on where the elevator will go if it must serve several requests for where to go. This "decision algorithm" should be clearly localized, so that it is easy to replace it by another one which is optimized for some particular purpose.

b) (15p) Equip your model with some animation, which shows the position of the elevator, the doors, and the state of buttons (lights). The user

should be able to control the elevator by "pushing buttons", according to the above description. You have freedom in how the animation looks; the animation does not have to be "fancy".

Save your solution to this exercise; you will use it again later.

Submission

Solutions (all files) to this assignment are to be submitted via the Student Portal by **Thursday, October 27, 2022**