Lecture Energy Systems: Hardware and Control - Control Part
University of Freiburg – Winterterm 2017/2018

Exercise 4: Discrete-Time State Space Control in MATLAB/Octave

Prof. Dr. Moritz Diehl, Dr. Gianluca Frison and Benjamin Stickan

The exercise is based on the scripts `ex2_octave.m` (Octave version) and `ex2_matlab.m` (Matlab version).

In this exercise, we will work with the mass-spring system. It consists on a chain of m masses (all of mass 1) connected each other with springs (all with spring constant 1). The first and the last mass are also connected to walls, through spring with spring constant 1. Forces can act on the first p masses.

This is a linear system described by the set of m second-order linear ordinary differential equations

$$\begin{aligned}
\ddot{q}_1 &= -2q_1 + q_2 + f_1 & \\
\ddot{q}_i &= q_{i-1} - 2q_i + q_{i+1} + f_i & i = 2, \dots, p \\
\ddot{q}_i &= q_{i-1} - 2q_i + q_{i+1} & i = p+1, \dots, m-1 \\
\ddot{q}_m &= q_{m-1} - 2q_m & i = m
\end{aligned}$$

where q_i is the deviation form the equilibrium point of the i-th mass and f_i is the force acting on the i-th mass.

This system can be transformed into a system of $2m$ first-order linear ordinary differential equations by introducing the velocities $v_i = \dot{q}_i$ for each mass

$$\begin{aligned}
\dot{q}_i &= v_i & i = 1, \dots, m \\
\dot{v}_1 &= -2q_1 + q_2 + f_1 & \\
\dot{v}_i &= q_{i-1} - 2q_i + q_{i+1} + f_i & i = 2, \dots, p \\
\dot{v}_i &= q_{i-1} - 2q_i + q_{i+1} & i = p+1, \dots, m-1 \\
\dot{v}_m &= q_{m-1} - 2q_m & i = m
\end{aligned}$$

We choose as state vector the vector $x(t) \in \mathbb{R}^{n_x}$ (with $n_x = 2m$) where the first m components are the deviations of the masses q_i and the last m components are the velocities of the masses v_i. The control vector $u(t) \in \mathbb{R}^{n_u}$ (with $n_u = p$) contains the forces f_i acting on the first p masses. We assume that we can measure the displacement and the velocity of each mass, and therefore the output of the system is equal to the state, $y(t) = x(t)$.

Using these definitions, the mass-spring system can be written as the state space system

$$\begin{aligned}
\dot{x}(t) &= Ax(t) + Bu(t) \\
y(t) &= Cx(t)
\end{aligned}$$

where

$$A_c = \begin{bmatrix} 0_{m\times m} & I_{m\times m} \\ T_{m\times m} & 0_{m\times m} \end{bmatrix}, \quad B_c = \begin{bmatrix} 0_{m\times p} \\ I_{p\times p} \\ 0_{(m-p)\times p} \end{bmatrix}, \quad C_c = I_{2m\times 2m}$$

where T is the $m \times m$ tridiagonal matrix

$$T = \begin{bmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & -2 & 1 \\ 0 & \dots & 0 & 1 & -2 \end{bmatrix}$$

In this exercise, we will choose a system with $m = 4$ masses and $p = 3$ forces. Therefore, the state-space system has size $n_x = 8$ and $n_u = 3$.

1. Is the continuous time system stable? Is it controllable? Why?

   Hint: you may want to use the commands `eig` and `rank`.

2. It is easy to discretize the continuous-time state space system by using the `c2d` command. In order to do so, you can create the continuous-time state space system by using the command `sys`, and afterwards pass it to the `c2d` command in order to get the discrete-time state space representation

$$\begin{aligned}
x_{k+1} &= Ax_k + Bu_k \\
y_k &= Cx_k
\end{aligned}$$

As sampling time, you can choose $T_s = 0.5$ seconds

Is the discrete time system stable? Is it controllable? Why?

3. Compute a closed-loop simulation of $N_s = 50$ steps of the discrete time system, starting with initial state

$$x_0 = \begin{bmatrix} 2.5 \\ 2.5 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

and using zero as control input at all steps.

Plot the result in 3 figures, one for the displacements, one for the velocities and one for the forces. What is the behaviour of the system? Is that expected? Why?

Hint: you may want to use the commands `dlqr` and `dare`.

4. Compute the gain K of the discrete time LQR regulator, choosing the penalty matrices $Q = I_{n_x \times n_x}$ and $R = 2 \cdot I_{n_u \times n_u}$. Compute the matrix $A - BK$ describing the dynamic of the closed-loop system. Is it stable?

Use K to compute the control input to use in the closed-loop simulation at each simulation step. Plot the result in 3 figures, one for the displacements, one for the velocities and one for the forces. What is the behaviour of the system? Is that expected? Why?

Now you can try to change the values of Q and R. How is the behaviour of the closed-loop system changing?

At each simulation step n, compute the performance index

$$J_n = \frac{1}{s} x_n^\top Q x_n + \frac{1}{2} u_n^\top R u_n$$

and then compute the cumulative closed loop cost

$$J_{N_s} = \sum_{n=0}^{N_s - 1} J_n$$

Is the value of J_{N_s} changing with N_s?

5. Now let's introduce saturation the control input, such that the control input takes value in the interval $u_{k,i} \in \begin{bmatrix} -0.5 & 0.5 \end{bmatrix}$.

Compute the input using LQR as in point 4., and then apply the saturation before using it in the simulation. Plot the result in 3 figures, one for the displacements, one for the velocities and one for the forces. What is the behaviour of the system? Is that expected? Why?

How does the closed loop cost change with respect to the one in point 4.?

6. In this point, you will use MPC to compute the control input to use in the closed-loop simulation. For now, you can fix the control horizon to $N = 2$. In MPC, the input saturation is part of the problem formulation, expressed as a constraint that is explicitly taken into account in the computation of the optimal control law.

Rewrite the MPC problem as a QP and use a QP solver (as e.g. `quadprog` in Matlab or `qp` in Octave) to solve it. Extract the input value at the first stage 0 and use it in the closed-loop simulation. Plot the result in 3 figures, one for the displacements, one for the velocities and one for the forces. Compare with point 5. for different values of N. What are the differences?

Plot the entire solution of the QP and compare it with the closed-loop simulation, and repeat for different values of N. What do you see?

Compute the closed loop cost and compare it with the value computed in 5. for different values of N. Is it larger or smaller?

Measure the average time needed to solve a QP during the simulation, and plot it for different values of N. What do you observe?