Model Predictive Control - EL2700

Assignment 2 : Finite-time optimal control through dynamic programming

2019

Automatic control
School of Electrical Engineering and Computer Science
Kungliga Tekniska Högskolan

Task: Optimal control through dynamic programming

In the first part of this assignment, we will design a finite-time optimal controller for stabilizing the inverted pendulum. In this task, we will disregard the dynamics of the cart and consider the simple dynamics of the inverted pendulum. To this end, you will apply a dynamic programming technique on the linear dynamics of the inverted pendulum. We will then extend the formulation to design a non-linear controller by feedback linearization. As a part of this task, you will fill in values in two MATLAB skeleton files, `assignment_2_linear_controller.m` and `assignment_2_nonlinear_controller.m`. You should upload the filled in MATLAB files to Canvas when you are finished. In addition, we ask you to analyze and reflect on some of the results. Write down these reflections in a report similar to the first assignment.

In the second part of this assignment, we will include cart dynamics and design a finite-time optimal controller for moving the cart from one position to the other while maintaining the pendulum upright. For this task, the necessary MATLAB files will be given to you. Your task will be to simulate the controller and compare it's performance to that of state feedback controller in the first design task.

Preparation As preparation for this assignment, work through exercise 3.17 in the compendium. This exercise introduces a numerical dynamic programming technique called gridding, which will be applicable in this assignment as well. Make sure you work through the whole exercise, with the solution manual if necessary, so that you become comfortable with using this technique.

Predefined source code As this exercise would be too large in its entirety we have predefined most of the required parameters and inline MATLAB functions required to solve the problems. These are given between '`=`' delimiters in the given skeleton MATLAB files and should NOT be edited. Your task will only revolve around defining cost functions and formulating dynamic programming procedures.

PART I: Dynamic programming for regulation

Inverted pendulum model We have modeled the continuous-time non-linear dynamics of the inverted pendulum using the following first order ODE system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -a_0 \sin x_1 - a_1 x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ b_0 u \cos x_1 \end{bmatrix} \tag{1}$$

where $x_1 = \theta$, and $x_2 = \dot{\theta}$ are states and cart acceleration u is control input. $a_0 = -\frac{mgl}{I+ml^2}$, $a_1 = \frac{b_p}{I+ml^2}$ and $b_0 = \frac{ml}{I+ml^2}$. We have implemented (1) in `inverted_pend.slc` SIMULINK model. Open the model and make sure you understand how it works.

Discrete linear model In the first design project, you linearized (1) around the vertical position and discretized the resulting linear system using zero-order-hold discretization with sample interval h. In this assignment, we will reuse the discrete-time linearized dynamics of the inverted pendulum model

$$x_{t+1} = Ax_t + Bu_t \tag{2}$$

where $x_t = [\theta \quad \dot{\theta}]^T \in \mathbb{R}^2$, $u_t \in \mathbb{R}$ are the system states and control input respectively.

Linear controller design In this section, you will use dynamic programming and a brute-force technique called gridding to stabilize the inverted pendulum on the cart. You will compute an optimal solution to the following simplified problem to balance the pendulum at it's upright position through cart acceleration

$$\begin{aligned} \underset{u_t}{\text{minimize}} \quad & \sum_{t=1}^{N} x_t^T Q x_t + u_t^T R u_t \\ \text{subject to} \quad & x_{t+1} = Ax_t + Bu_t \quad k = 1, \ldots, N-1 \end{aligned} \tag{3}$$

Using dynamic programming, we can find the optimal control u_t by solving the one-step optimization problem

$$J_t(x_t) = \min_{u_t} x_t^T Q x_t + u_t^T R u_t + J_{t+1}(Ax_t + Bu_t). \tag{4}$$

The difficulty lies in calculating the cost-to-go function $J_{t+1}(\mathbf{x}_{t+1})$. We can approximate the cost-to-go by gridding the state-space and calculating the value function at these grid points. We will use the MATLAB functions `fminunc` and `interp2` to calculate the cost-to-go.

To this end, you will tune penalty matrices Q and R to balance the level of stabilization with the size of the control input. Your task is to implement an appropriate dynamic programming procedure by filling in `assignment_2_linear_controller.m`. Finally, simulate your controller with the initial state $x_0 = [\pi/6 \quad 0]^T$. You can modify the initial state in `prepare_sim.m`. Do not forget to set `DP_TYPE` $= 0$ in `finite_time_DP.m` and supply penalty matrices Q, and R in `post_sim.m` to compute accumulated cost and input energy.

Non-linear controller design One advantage of gridding is that we can include the system non-linearities in the controller design. An alternative approach to linearizing the inverted pendulum model is a technique called feedback linearization. Lets redefine the input as

$$u = \tfrac{1}{\cos x_1}\Big(\tfrac{a_0}{b_0}\big(\sin x_1 - x_1\big) + v\Big). \tag{5}$$

Then, the non-linear dynamics of the inverted pendulum (1) become linear for the fictitious control input v

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ b_0 \end{bmatrix} v \tag{6}$$

To design v, we need to substitute the expression for u in (5) into the stage-cost

$$J_{stage}(x_t, v_t) = x_t^T Q x_t + R\left[\tfrac{1}{\cos x_t(1)}\Big(\tfrac{a_0}{b_0}\Big(\sin x_t(1) - x_t(1)\Big) + v\Big)\right]^2 \tag{7}$$

We can then find the optimal $v^\star$ by solving

$$J_t(x_t) = \min_{v_t} J_{stage}(x_t, v_t) + J_{t+1}(Ax_t + Bv_t) \tag{8}$$

The optimal control $u^\star$ is then given by (5). Your task is to implement the non-linear controller in (5) by modifying the stage cost J_{stage} according to (7) and incorporating the fictitious control input v in `assignment_2_nonlinear_controller.m`. You can use the same penalty matrices as in the linear formulation to be able to compare the performance of this controller to its linear counterpart. Now, simulate your controller with the same initial point. Do not forget to set `DP_TYPE` $= 1$ in `finite_time_DP.m`. Convergence can be improved by using the infinite-horizon LQR cost as your initial cost-to-go, using the `dlqr` command. We will discuss this further in the coming weeks.

Performance comparison You will now compare the performance of the linear formulation and the non-linear formulation in terms of the following performance measures:

1. computation time for offline calculation of the optimal control for same grid size
2. accumulated cost and spent control energy
3. convergence rate to the upright position

Compare the performance of the two formulations and summarize your findings in the report.

As we have used a linearized pendulum model around the vertical position in the formulation of the linear controller, we can expect that the resulting model will not be accurate if the initial position is far from the upright position. To validate this, you will now simulate both controllers with the following initial states:

1. $x_0 = [\pi/4 \quad 0]^T$
2. $x_0 = [\pi/3 \quad 0]^T$

Are both controllers working satisfactory for both initial positions? Analyze your result and summarize in the report.

PART II: Dynamic programming for reference tracking

Dynamic programming is a powerful technique for solving optimal control problems. However, it suffers from the notorious "curse of dimensionality", which prevents its direct application when the dimension of the state space is high. To observe this problem, we will now consider a four dimensional state-space model where we include both cart and pendulum dynamics.

Linear controller design We will now design a finite time optimal controller for moving the cart from one position to another while keeping the pendulum upright. We will reuse the linearized cart-pendulum model around the upright position which you derived in the first design project. The discrete-time linear dynamics of the cart pendulum model can be represented as:

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + Bu_t \tag{9}$$

where $\mathbf{x}_t = [x \quad \dot{x} \quad \theta \quad \dot{\theta}]^T \in \mathbb{R}^4$, $u_t \in \mathbb{R}$ are the system states and control input respectively. We will now compute an optimal solution to the following tracking problem

$$\begin{aligned} \underset{u_t}{\text{minimize}} \quad & \sum_{t=1}^{N} (\mathbf{x}_t - \mathbf{x}_r)^T Q (\mathbf{x}_t - \mathbf{x}_r) + u_t^T R u_t \\ \text{subject to} \quad & \mathbf{x}_{t+1} = A\mathbf{x}_t + Bu_t \quad k = 1, \dots, N-1 \end{aligned} \tag{10}$$

where $\mathbf{x}_r$ is the reference state we want to track. In this problem, $\mathbf{x}_r = [10 \quad 0 \quad 0 \quad 0]^T$. To solve this problem, we have constructed a five dimensional grid corresponding to four system states and the time dimension in `assignment_2_TVFS.m`. We have used the builtin MATLAB functions `fminunc` and `interpn` to calculate the cost-to-go. Open this MATLAB script and make sure you understand how we have implemeted (10) using dynamic programming through gridding. We encourage you to run this script and generate `DP_TV_L_FS.mat` by your own. However, keep in mind that it takes approximately 2 hours to generate the `DP_TV_L_FS.mat` file. Therefore, we have also provided a `DP_TV_L_FS.mat` file to you in case you fail to generate your own. After running the simulation you can run `animation.m` and observe the movement of the pendulum. We hope that you will appreciate the "curse of dimensionality" when using this approach. Later in the course, we will show how model predictive control can alleviate this issue without a significant loss of performance.

Now, simulate the controller with an initial state $\mathbf{x}_0 = [0 \quad 0 \quad \pi/6 \quad 0]^T$. Compare the tracking performance of this controller with the state feedback regulator you have designed in the previous project. Write down your reflections in the report.

PART III: Submitting MATLAB files

To complete this design project, you should upload a small report with your reflections and filled in MATLAB skeleton files:

- `assignment_2_linear_controller.m`
- `assignment_2_nonlinear_controller.m`

to Canvas.

Good Luck!