April 17, 2024

6.3100 Lecture 19 Notes – Spring 2024

Linear quadratic regulator (LQR) control

Dennis Freeman, Elfar Adalsteinsson, and Kevin Chen

Outline:

1. LQR formulation and solution
2. Example: inverted pendulum
3. Controllability

1. LQR formulation and solution

We start the LQR formulation with the state space form:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

where $u, y, and\ x$ are the input, output, and state variables. The control input u(t) is given by:

$$u(t) = K_r r - Kx$$

In the previous lecture, we described the pole placement method where we can set the eigenvalues of A-BK by designing the K matrix. The method was straightforward, but not very intuitive for control designers. Our goal is to come up with an intuitive control method.

To design the control matrix K, we define a quadratic cost function:

$$J = \int_0^\infty (x^T Q x + u^T R u)\, dt$$

where Q is a $n \times n$ matrix (n is the size of state vector x) and R is a $m \times m$ matrix (m is the size of input variable u). In this course, we only consider single-input-single-output systems, so m is 1, and R is a scalar. To ensure the cost function J is a positive number for all x(t), we impose that:

$$Q \geq 0, and\ R > 0$$

$Q \geq 0$ means it is a semi positive definite matrix, and R>0 means it is a positive number. In simple terms, $Q \geq 0$ implies $x^T Q x \geq 0$ for all x. Hence, the function J is always positive.

The LQR control method calculates a matrix K that minimizes the total cost J. Solving this matrix K involves the Ricatti equation, which is a complex process involving advanced math tools. We simply state the result here. The optimized matrix K is given by:

$$K = R^{-1}(B^T P(t))$$

where the matrix P is the solution to the equation:

$$A^T P + PA - (PB)R^{-1}(B^T P) + Q = 0$$

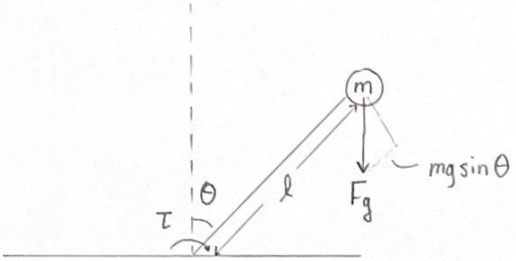
This equation is quadratic in P and it is difficult to be solved analytically. We will use computational tools such as MATLAB.

The key design question here is how to choose the matrices Q and the scalar R. The design process is intuitive because Q penalizes the system convergence speed, and R penalizes the control effort. If Q is large, then the system would converge faster. In contrast, if Q is small compared to R, then the system would converge slower with a smaller control effort. The MATLAB syntax is shown below:

$$K = lqr(A, B, Q, R)$$

2. Example: inverted pendulum

Let's visit the inverted pendulum example again. The free-body diagram is shown below:



In this problem, the input is commanding torque τ, the output is the measured angle θ, and the states are $[\theta, \dot{\theta}]$. The system Newton second law is given by:

$$ml^2\ddot{\theta} = mgl\sin\theta + \tau$$

We linearize the equation to get:

$$\ddot{\theta} \approx \frac{g}{l}\theta + \frac{1}{ml^2}\tau$$

We let $x = [\theta, \dot{\theta}]$, and write the problem in state space form:

$$\frac{d}{dt}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 1 \\ g/l & 0\end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0 \\ 1/ml^2\end{bmatrix}\tau$$

We want to balance the system at equilibrium, so our input command τ is set to:

$$\tau = K_r r - Kx = -Kx = -[k_1\ k_2]\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix}$$

The output is given as:

$$y = \theta = [1\ 0]\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} + 0 \times \tau$$

We are going to solve this problem in MATLAB. First, we define the problem parameters:

```
%define system parameters
g = 9.8; l = 1; m = 0.1 ;

%form A and B matrices
A = [0 1; g/l 0];
B =[0;1/(m*l^2)];
```

Next, we define the Q and R matrices. Note that scaling Q and R by a constant number always give the same results. That is, for example, if choosing (Q*, R*) gives the matrix K*, then choosing (5Q*, 5R*) also give the matrix K*. To normalize the system, we always set R = 1.

We will compare two cases by choosing two weight matrices Q1 and Q2:

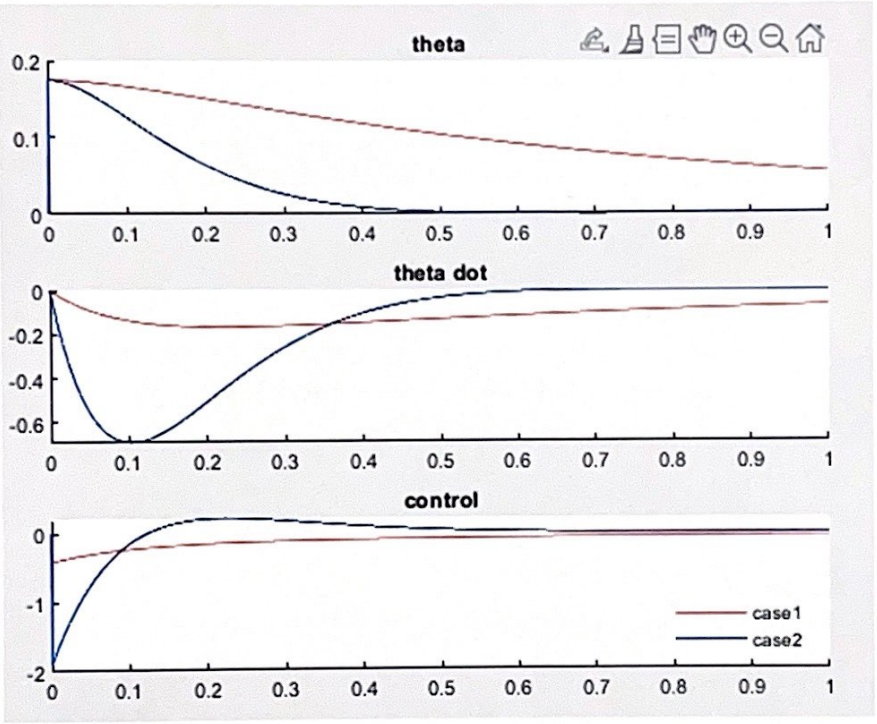
$$Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, Q_2 = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$$

Intuitively, Q1 has a smaller weight on the states while Q2 has a larger weight on the states. We would expect the solution from Q1 will converge slower with a gentler input, while the solution from Q2 will converge faster with a more aggressive input. Note the off-diagonal terms are 0, which implies there is no off-axis coupling. In almost all design cases, Q is a diagonal matrix with positive numbers in its diagonal entries. The code for setting up the problem and solving the problem is given below:

```
%define Q and R matrices
Q1 = diag([1 1]); R = 1;
Q2 = diag([100 1]); R = 1;

%form 2 control matrices
K1 = lqr(A,B,Q1,R);
K2 = lqr(A,B,Q2,R);

%solve the problem
x0 = [10 *(pi/180); 0];
t_vec = linspace(0,1,1000);
x1 = zeros(2,length(t_vec)); x2 = x1;
c1 = zeros(size(t_vec));     c2 = c1;
control_m1 = A-B*K1; control_m2 = A-B*K2;
x(:,1) = x0;
for i = 2:length(t_vec)
    x1(:,i) = expm(control_m1*t_vec(i))*x0;
    c1(i) = - K1 * x1(:,i);

    x2(:,i) = expm(control_m2*t_vec(i))*x0;
    c2(i) = - K2 * x2(:,i);
end
```

Finally, we can plot the solutions. Here, we show both the states and the control input. The code and the corresponding graphs are shown below.

```
%plot
close all
figure(1)
subplot(3,1,1); title('theta'); hold on
plot(t_vec,x1(1,:),'r-'); plot(t_vec,x2(1,:),'b-');

subplot(3,1,2)
title('theta dot'); hold on
plot(t_vec,x1(2,:),'r-'); plot(t_vec,x2(2,:),'b-');

subplot(3,1,3)
title('control'); hold on
plot(t_vec,c1,'r-'); plot(t_vec,c2,'b-');
axis([0 1 -0.03 0.0005]);
legend('case1','case2','location','southeast'); legend boxoff
```



Based on the simulation result, we see that case 1 converges to 0 much slower, and its control effort is low. Case 2 converges to 0 much faster, but its control efforts vary over a large range. This design method is a lot more intuitive than other methods.

3. Controllability

Here, we introduce an advanced concept called controllability. Some physical systems may not be stabilizable regardless of how we design our controllers. Can we always place every eigenvalue to our desired location? It turns out the property of the A and B matrices determine whether we can fully control a system.

Definition: A control system is **controllable** if the state vector x(t) can be driven to any desired values from an arbitrary initial condition x0.

Result: A system is **controllable** if rank([B AB A^2B ... $A^{n-1}B$])=n, where n is the size of x.

Proof: (this proof is optional for this course)

The general solution to a state-space system is given by:

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

We need to expand $e^{A(t-\tau)}$. We let $\tau' = t - \tau$, and this gives:

$$e^{A\tau'} = I + (A\tau') + \frac{1}{2!}(A\tau')^2 + \frac{1}{3!}(A\tau')^3 + \cdots$$

Next, we apply the Cayley-Hamilton theorem:

$$\begin{aligned} \det(\lambda I - A) = 0 \quad &= \quad \lambda^n + \alpha_{n-1}\lambda^{n-1} + \cdots + \alpha_1\lambda + \alpha_0 \\ &= \quad A^n + \alpha_{n-1}A^{n-1} + \cdots + \alpha_1 A + \alpha_0 \end{aligned}$$

This implies:

$$A^n = -(\alpha_{n-1}A^{n-1} + \cdots + \alpha_1 A + \alpha_0)$$

And this condition implies:

$$e^{A\tau'} = \beta_0 + \beta_1\tau' A + \beta_2\tau'^2A^2 + \cdots + \beta_{n-1}\tau'^{n-1}A^{n-1}$$

The key takeaway is $e^{A\tau'}$ is expressed as a sum of n terms. Next, we write the general solution again:

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

Through change of variables, we have

$$x(t) = e^{At}x_0 + \int_0^t e^{A\tau'}Bu(t-\tau')d\tau'$$

$$x(t) = e^{At}x_0 + \int_0^t (\beta_0 + \beta_1\tau' A + \beta_2\tau'^2A^2 + \cdots + \beta_{n-1}\tau'^{n-1}A^{n-1})Bu(t-\tau')d\tau'$$

$$x(t) = e^{At}x_0 + \sum_{i=0}^{n-1} \int_0^t \beta_i(\tau')^i A^i B u(t-\tau') d\tau'$$

We can define $\gamma_i(t) = \int_0^t \beta_i(\tau')^i u(t-\tau') d\tau'$

The expression becomes:

$$x(t) = e^{At}x_0 + \sum_{i=0}^{n-1} A^i B \gamma_i(t)$$

$$x(t) = e^{At}x_0 + [B\ AB\ A^2B \dots A^{n-1}B] \begin{bmatrix} \gamma_0(t) \\ \gamma_1(t) \\ \gamma_2(t) \\ \dots \\ \gamma_{n-1}(t) \end{bmatrix}$$

To reach every possible state x(t), the matrix $[B\ AB\ A^2B \dots A^{n-1}B]$ needs to have full rank.