Christopher Lum
lum@uw.edu

Lecture 10a
Introduction to Linear Quadratic Regulator (LQR) Control

Lecture is on YouTube

YouTube video entitled 'Introduction to Linear Quadratic Regulator (LQR) Control' covering this is located at https://youtu.be/wEevt2a4SKI.

Outline

Introduction

As we saw previously, full state feedback is a powerful control law but it is difficult to judge how the placement of the closed loop poles will affect the control law (https://youtu.be/9vCTokJ5RQ8). In general, we saw that aggressive pole placement (ie closed loop poles are located far from the open loop poles) requires a large control signal. It would be useful if we had a tool we could use to trade off between performance (ie $\overline{x}(t) \to \overline{0}$ quickly) vs. the required control (relative size of $\overline{u}(t)$ required to regulate states to zero). This is the basis of the linear-quadratic regulator (LQR).

This method takes an optimization approach to the problem. Optimization is one of the most important branches of mathematics for control systems. This functions by using a mathematical function to measure the cost of a solution and simultaneously checking if this solution is feasible with respect to a set of constraints. Both the cost function and constraints must be carefully setup in order to generate reasonable solutions.

A Brief Introduction to Optimization

Consider the following optimization problem. We would like to figure out the best way to commute from Bainbridge Island to the UW campus using an optimization framework. Consider a set of possible solutions as

z_1 = drive from house to ferry, catch ferry, drive to campus
z_2 = bus from house to ferry, catch ferry, bus to campus
z_3 = bike from house to ferry, catch ferry, bike to campus
z_4 = run to shoreline, swim across Puget sound, run to campus
z_5 = rent helicopter to fly from house to campus directly

We can denote the set of all possible solutions as $Z = \{z_1, z_2, \ldots, z_5\}$. This is often referred to as the feasible set

$Z = \{z_1, z_2, \ldots, z_5\}$ = feasible set

We now need a metric which will measure the "goodness" of each solution. Often this is called a cost function

$J(z)$ = cost function

<u>Situation 1: Minimize Time</u>

Suppose that we would like to minimize the amount of time required for the commute. Therefore, the cost function would be

$J_1(z)$ = time required for action z

Therefore, the optimization problem can be formulated as

$(\wp_1)\ \underset{z \in Z}{\text{minimize}}\ J_1(z)$ (minimize cost function $J_1(z)$ subject to z must be in the feasible set Z)

In this situation, the solution to this problem, $(\wp_1)$, would be

z_5 = rent helicopter to fly from house to campus directly

This would most likely get you to campus the fastest. However, this would cost a significant amount of money.

<u>Situation 2: Minimize Money</u>

The problem with cost function $J_1(z)$ is that we did not include any monetary aspects into the cost function. Therefore, another potential cost function might be

$J_2(z)$ = money required for action z

Therefore, the optimization problem can be formulated as

$$(\wp_2)\ \underset{z \in Z}{\text{minimize}}\ J_2(z)$$

In this situation, the solution to this problem, $(\wp_2)$, would most likely be

z_4 =run to ferry, swim across Puget sound, run to campus

This would most likely get you to campus the and spend the least amount of money. However, this would result in an unrealistic time to commute (and other factors such as likely death swimming 5 miles across Puget sound).

<u>Situation 3: Minimize Time/Money Tradeoff</u>

So a 3rd potential cost function might be a tradeoff between time and money. Therefore, another potential cost function might be

$$J_3(z) = q\,J_1(z) + r\,J_2(z)$$

where q, r = scalar coefficients to tradeoff between time and money

Therefore, the optimization problem can be formulated as

$$(\wp_3)\ \underset{z \in Z}{\text{minimize}}\ J_3(z)$$

So we see that we can tune the optimization problem by choosing appropriate q and r parameters. For example

large q, small $r \Rightarrow$ problem reverts to a minimize time problem $\Rightarrow z_5$ = helicopter
small q, large $r \Rightarrow$ problem reverts to a minimize money problem $\Rightarrow z_4$ = swim
medium q, medium $r \Rightarrow$ tradeoff between time and money compromise $\Rightarrow z_1$, z_2, or z_3 depending on q and r

<u>Situation 4: Minimize Time/Money Tradeoff with Additional Constraints</u>

We can add additional constraints to the problem to further tailor the behavior of the solution. For example, let us continue with our 3rd cost function (tradeoff between time and money) but add an additional constraint to the problem. For example, we could require that during the commute, we would like to get at least 30 minutes of exercise as well.

constraint 1 = commute yields at least 30 minutes of exercise

Therefore, the optimization problem can be formulated as

$$(\wp_4)\ \underset{z \in Z}{\text{minimize}}\ J_3(z)$$

such that $f_1(z) \geq 30$

where $f_1(z)$ = number of minutes exercised during action z

This additional constraint has the effect of making the feasible set smaller.

$f_1(z_1) \ngeq 30 \Rightarrow z_1$ is not allowed as a solution (drive from house to ferry, catch ferry, drive to campus)
$f_1(z_2) \ngeq 30 \Rightarrow z_2$ is not allowed as a solution (bus from house to ferry, catch ferry, bus to campus)
$f_1(z_3) \geq 30 \Rightarrow z_3$ is a potential valid solution (bike from house to ferry, catch ferry, bike to campus)
$f_1(z_4) \geq 30 \Rightarrow z_4$ is a potential valid solution (run to ferry, swim across Puget sound, run to campus)
$f_1(z_5) \ngeq 30 \Rightarrow z_5$ is not allowed as a solution (rent helicopter to fly from house to campus directly)

Therefore, we could specify another feasible set as

$$\tilde{Z} = \{z_3, z_4\}$$

Then problem ($\wp_4$) could be alternatively expressed as

$$(\wp_4) \underset{z=\tilde{Z}}{\text{minimize}} \, J_3(z)$$

In this situation, the solution to this problem, ($\wp_4$), would most likely be

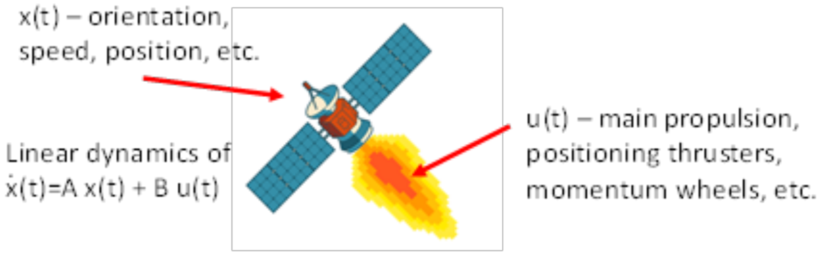
z_3 = bike from house to ferry, catch ferry, bike to campus) (satisfies exercise constraint, good tradeoff between time/money)

The Linear Quadratic Regulator

We can apply this idea of optimization to the problem of designing a controller for a linear, dynamic system.

It may be helpful to consider a physical system for the remainder of the discussion. For example, consider a satellite in space. In this scenario

$$\overline{x}(t) = \begin{pmatrix} \text{orientation} \\ \text{speed} \\ \text{position} \\ \ldots \end{pmatrix} \qquad \overline{u}(t) = \begin{pmatrix} \text{main propulsion} \\ \text{positioning thrusters} \\ \text{momentum wheels} \\ \ldots \end{pmatrix}$$


x(t) – orientation, speed, position, etc.
Linear dynamics of ẋ(t)=A x(t) + B u(t)
u(t) – main propulsion, positioning thrusters, momentum wheels, etc.

Setting Up the Optimization Problem

Consider the cost function of the form

$$J = \int_0^\infty \overline{x}(t)^T Q \overline{x}(t) + \overline{u}(t)^T R u(t) \, dt \qquad \textbf{(Eq.1)}$$

where $\overline{x}(t) = n \times 1$ state vector

$\overline{u}(t) = m \times 1$ control vector

$Q = n \times n$ symmetric, positive semidefinite matrix (written as $Q \geq 0$)

$R = m \times m$ symmetric, positive definite matrix (written as $R > 0$)

The optimization problem in this context is

$$(\wp) \ \underset{\overline{u}(t) \in \mathbb{R}^m}{\text{minimize}} \ J = \int_0^\infty \overline{x}(t)^T Q \overline{x}(t) + \overline{u}(t)^T R \overline{u}(t) \, dt \qquad \textbf{(Eq.2)}$$

$$\text{such that } \dot{\overline{x}}(t) = A \overline{x}(t) + B \overline{u}(t)$$

A positive semidefinite matrix, Q, has the definition that

$\overline{x}(t)^T Q \overline{x}(t) \geq 0 \ \forall \overline{x}(t)$ (this quantity is greater than or equal 0 for all possible $\overline{x}(t)$)

Likewise, a positive definite matrix, R, has the definition that

$\overline{u}(t)^T R \overline{u}(t) > 0 \ \forall \overline{u}(t)$ (this quantity is strictly greater than 0 for all possible $\overline{u}(t)$)

Therefore, if $\overline{x}(t)$ or $\overline{u}(t)$ is non-zero the integrand will be a positive value. Since the cost function is the integral of $\overline{x}(t)^T Q \overline{x}(t) + \overline{u}(t)^T R u(t)$ over time, if $\overline{x}(t)$ or $\overline{u}(t)$ remain non-zero, the cost function will continue to increase. Therefore, the best thing to do is the simply have $\overline{x}(t)$ and $\overline{u}(t)$ be zero as quickly as possible. However, this involves a tradeoff because in order to drive the states to zero, you typically need to exercise some control authority. In the case of our satellite, if you want to return the orientation of the satellite to the origin, then you need to spend some fuel/propulsion to do so.

We can use the matrices Q and R to tradeoff between the importance of the state going to zero quickly (the $\overline{x}(t)^T Q \overline{x}(t)$ term) and how much control is required to drive the state to zero (the $\overline{u}(t)^T R \overline{u}(t)$ term).

Roughly speaking, if the Q matrix is large compared to the R matrix, then it costs more to have a non-zero state (the $\int_0^\infty \overline{x}(t)^T Q \overline{x}(t) \, dt$ dominates). Therefore, it is better to use a large amount of control to quickly drive the state to zero.

Conversely, if R is large compared to Q, then it costs more to use any control (the $\int_0^\infty \overline{u}(t)^T R\,\overline{u}(t)\,dt$ dominates). Therefore, it is better to use a small amount of control to slowly regulate the states back to zero.

In general

$$\{Q \text{ "bigger than" } R\} \Rightarrow \left\{\text{fast regulation of } \overline{x}(t) \to \overline{0},\ \overline{u}(t) \text{ is large}\right\}$$
$$\{R \text{ "bigger than" } Q\} \Rightarrow \left\{\text{slow regulation of } \overline{x}(t) \to \overline{0},\ \overline{u}(t) \text{ is small}\right\}$$

we use quotes because as we see, Q and R are not the same dimension and how they influence the optimization problem depends on how the states and controls are defined.

We can use a simple example of 2 states and 2 controls to illustrate the cost function

$$\overline{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \qquad \overline{u}(t) = \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}$$

```
x = (x1
     x2); u = (u1
               u2);
```

Consider a Q and R matrix as simple diagonal matrices

$$Q = \begin{pmatrix} q_{11} & 0 \\ 0 & q_{22} \end{pmatrix} \qquad R = \begin{pmatrix} r_{11} & 0 \\ 0 & r_{22} \end{pmatrix}$$

```
Q = (q11   0
      0  q22); R = (r11   0
                     0  r22);
```

So the integrand of the cost function is given as

```
integrand = (Transpose[x].Q.x + Transpose[u].R.u)[[1, 1]]
Clear[x, u, Q, R]
```

```
r11 u1^2 + r22 u2^2 + q11 x1^2 + q22 x2^2
```

So we see that

$$J = \int_0^\infty q_{11}\, x_1(t)^2 + q_{22}\, x_2(t)^2 + r_{11}\, u_1(t)^2 + r_{22}\, u_2(t)^2\, dt$$

We can now directly interpret the weighting matrix Q and R

q_{11} = weight/penalty on a non-zero $x_1(t)$
q_{22} = weight/penalty on a non-zero $x_2(t)$
r_{11} = weight/penalty on a non-zero $u_1(t)$
r_{22} = weight/penalty on a non-zero $u_2(t)$

Solving the Optimization Problem

The solution to ($\wp$) (which is beyond the scope of this class, see AA548) is given as

$$\overline{u}(t) = -K\,\overline{x}(t) \qquad \textbf{\textit{(Eq.3)}}$$

where $K = R^{-1} B^T S$

and S is the solution to the Algebraic Riccati equation (ARE).

$$A^T S + S A - S B R^{-1} B^T S + Q = 0 \qquad \textbf{\textit{(Eq.4)}}$$

note that S is an n-by-n symmetric matrix.

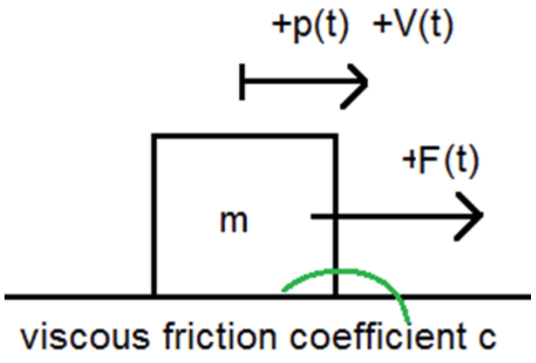
So we see that an LQR controller is simply a full state feedback controller with the gain K computed from the solution of a carefully constructed optimization problem.

So the workflow for designing an LQR controller is

1. Given A and B (plant).
2. Chose Q and R.
3. Solve ARE for S (yields multiple solution for S).
4. Compute $K = R^{-1} B S$ (yields multiple solutions for K).
5. Chose K that yields a stable system (all eig($A - B K$) < 0).

Example: Simple mass/damper system

Consider the mass and damper system shown below



Equations of motion for this system are

$$m\,\ddot{p}(t) = F(t) - c\,V(t)$$

where $p(t)$ = position
$V(t)$ = velocity
$F(t)$ = external force
m = mass of block
c = viscous damping coefficient of block

We can choose a state and control vector of

$$\overline{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} p(t) \\ V(t) \end{pmatrix} \qquad \overline{u}(t) = (u_1(t)) = (F(t))$$

So the state space representation of this system is given as

$$\dot{\overline{x}}(t) = \begin{pmatrix} 0 & 1 \\ 0 & -c/m \end{pmatrix} \overline{x}(t) + \begin{pmatrix} 0 \\ 1/m \end{pmatrix} \overline{u}(t)$$

Let us consider a case where $m = 1$, $c = 0.2$. So numerically we have

$$A = \begin{pmatrix} 0 & 1 \\ 0 & -1/5 \end{pmatrix} \; B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Furthermore, let us choose a Q and R matrix of

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad R = (0.01)$$

```
(*Step 1: Define A and B*)
m = 1;
c = 2 / 10;
A = (0   1
     0  -c / m);
B = (  0
     1 / m);

(*Step 2: Chose Q and R*)
Q = (1  0
     0  1);
R = {{1 / 100}};
```

We can attempt to solve the ARE of

$$A^T S + S A - S B R^{-1} B^T S + Q = 0$$

We make use of the fact that S is symmetric

```
(*Step 3: Solve ARE*)
```
$$S = \begin{pmatrix} s11 & s12 \\ s12 & s22 \end{pmatrix};$$

```
LHS = Transpose[A].S + S.A - S.B.Inverse[R].Transpose[B].S + Q;
LHS // MatrixForm
```

$$\begin{pmatrix} 1 - 100\, s12^2 & s11 - \frac{s12}{5} - 100\, s12\, s22 \\ s11 - \frac{s12}{5} - 100\, s12\, s22 & 1 + 2\, s12 - \frac{2\, s22}{5} - 100\, s22^2 \end{pmatrix}$$

We can now attempt to solve the ARE

```
temp = Solve[{LHS[[1, 1]] == 0, LHS[[1, 2]] == 0, LHS[[2, 2]] == 0}, {s11, s12, s22}]
```

$$\left\{\left\{s11 \to \frac{\sqrt{2001}}{50},\ s12 \to -\frac{1}{10},\ s22 \to \frac{1}{500} \times \left(-1 - \sqrt{2001}\right)\right\},\right.$$
$$\left\{s11 \to -\frac{\sqrt{2001}}{50},\ s12 \to -\frac{1}{10},\ s22 \to \frac{1}{500} \times \left(-1 + \sqrt{2001}\right)\right\},$$
$$\left\{s11 \to -\frac{\sqrt{3001}}{50},\ s12 \to \frac{1}{10},\ s22 \to \frac{1}{500} \times \left(-1 - \sqrt{3001}\right)\right\},$$
$$\left.\left\{s11 \to \frac{\sqrt{3001}}{50},\ s12 \to \frac{1}{10},\ s22 \to \frac{1}{500} \times \left(-1 + \sqrt{3001}\right)\right\}\right\}$$

We see this yields 4 unique solutions

```
solution1 = temp[[1]];
solution2 = temp[[2]];
solution3 = temp[[3]];
solution4 = temp[[4]];

S1 = S /. solution1;
S2 = S /. solution2;
S3 = S /. solution3;
S4 = S /. solution4;

S1 // MatrixForm // N
S2 // MatrixForm // N
S3 // MatrixForm // N
S4 // MatrixForm // N
```

$$\begin{pmatrix} 0.894651 & -0.1 \\ -0.1 & -0.0914651 \end{pmatrix}$$

$$\begin{pmatrix} -0.894651 & -0.1 \\ -0.1 & 0.0874651 \end{pmatrix}$$

$$\begin{pmatrix} -1.09563 & 0.1 \\ 0.1 & -0.111563 \end{pmatrix}$$

$$\begin{pmatrix} 1.09563 & 0.1 \\ 0.1 & 0.107563 \end{pmatrix}$$

Now we can compute the control gain $K = R^{-1} B^T S$

```
(*Step 4: Compute K*)
K1 = Inverse[R].Transpose[B].S1;
K2 = Inverse[R].Transpose[B].S2;
K3 = Inverse[R].Transpose[B].S3;
K4 = Inverse[R].Transpose[B].S4;

K1 // MatrixForm // N
K2 // MatrixForm // N
K3 // MatrixForm // N
K4 // MatrixForm // N
```

$$\begin{pmatrix} -10. & -9.14651 \end{pmatrix}$$

$$\begin{pmatrix} -10. & 8.74651 \end{pmatrix}$$

$$\begin{pmatrix} 10. & -11.1563 \end{pmatrix}$$

$$\begin{pmatrix} 10. & 10.7563 \end{pmatrix}$$

As we see, we obtain many solutions to this but we know that we want the solution that will be a stabilizing solution. We know the closed loop eigenvalues are given by eig($A - B K$)

```
(*Step 5:  Find solution that yields stable closed loop system*)
λ1 = Eigenvalues[A - B.K1];
λ2 = Eigenvalues[A - B.K2];
λ3 = Eigenvalues[A - B.K3];
λ4 = Eigenvalues[A - B.K4];

λ1 // N
λ2 // N
λ3 // N
λ4 // N
{9.95139, -1.00488}

{-9.95139, 1.00488}

{9.95139, 1.00488}

{-9.95139, -1.00488}
```

So we see that only the 4th solution yields a stable system so we use this as the solution

$$K = \begin{pmatrix} 10 & 10.76 \end{pmatrix}$$

Luckily for us, Matlab provides the function “lqr” to solve this

$$[K,\ S,\ E] = \text{lqr}(A, B, Q, R)$$

where $\quad E = \text{eig}(A - B\,K)$ (closed loop eigenvalues)

We investigate this now with a numerical example

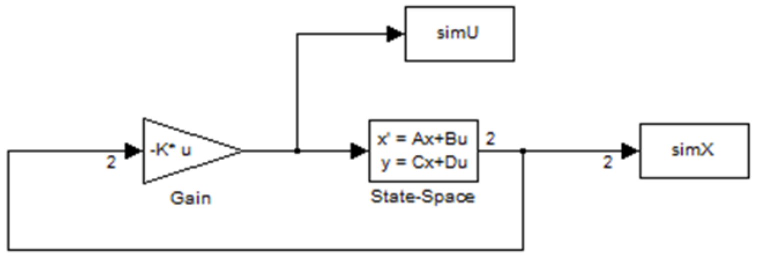
Example: Effect of Q and R on Solution

Consider the same mass/damper system we investigated previously.

Let us consider 3 scenarios.

Scenario 1: control is cheap and a non-zero state vector is expensive
Scenario 2: control is expensive and a non-zero state vector is cheap
Scenario 3: only a non-zero velocity is expensive

We can use LQR to design controllers (a matrix K) for these three situations. We can simulate the system with a non-zero initial condition of $\overline{x}(0) = \begin{pmatrix} \pi & -2 \end{pmatrix}^T$ and verify that the states are regulated to zero using the Simulink model shown below.



Scenario 1: Cheap Control

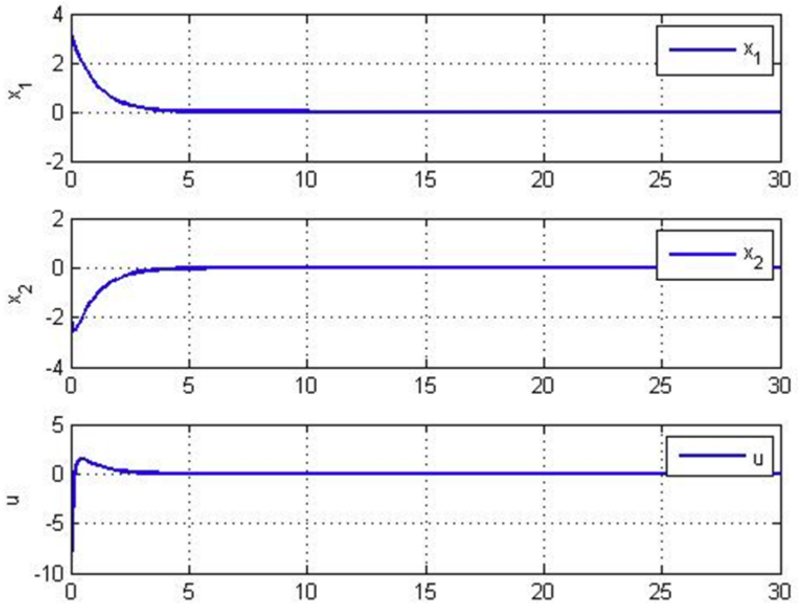
In this situation, the control is cheap but a non-zero state vector is expensive. Therefore, we want a *Q* matrix which is relatively larger than the *R* matrix. One candidate to try is

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad R = (0.01)$$

We use the LQR method to design a controller using these matrices. This yields

$$K = \begin{pmatrix} 10 & 10.76 \end{pmatrix}$$

The results of the system under this control law are shown below



As can be seen, the controller exerts a lot of control to quickly drive the system state to 0. The system states approach zero relatively quickly (less than 5 seconds)

Scenario 2: Expensive Control

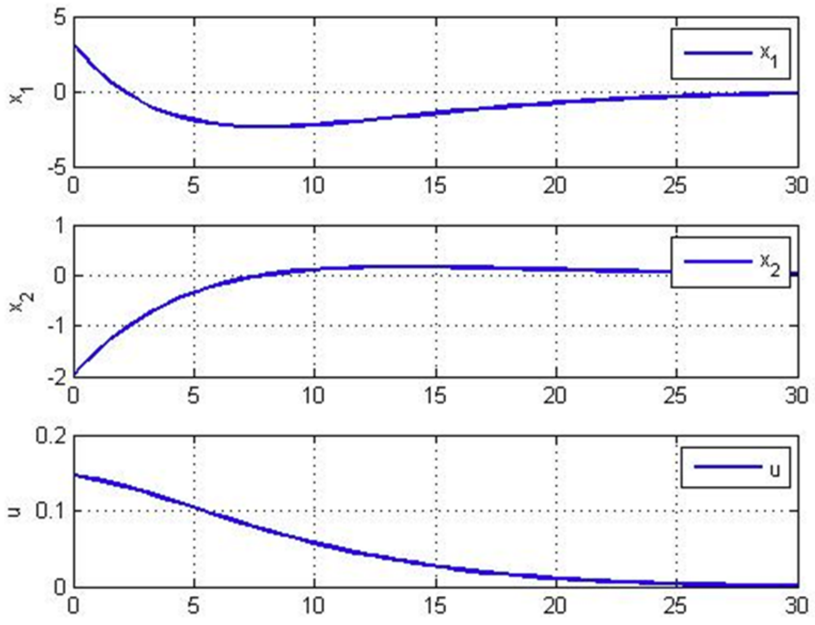
In this situation, the control is expensive but a non-zero state vector is cheap. Therefore, we want a *Q* matrix which is relatively smaller than the *R* matrix. One candidate to try is

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad R = (1000)$$

This yields

$$K = (\ 0.03 \quad 0.12\ )$$

The results of the system under this control law are shown below



As can be seen, the controller does not exert a lot of control in an attempt to keep the $\int_0^\infty \overline{u}(t)^T R\, \overline{u}(t)\, dt$ term small. As a result of this, the system state is not driven to zero. Notice that it takes almost 30 second before the system reaches zero (compare to case 1 where it took less than 5 seconds to reach $\overline{x}(t) \to \overline{0}$)

Scenario 3: Only Non-Zero Velocity is Expensive

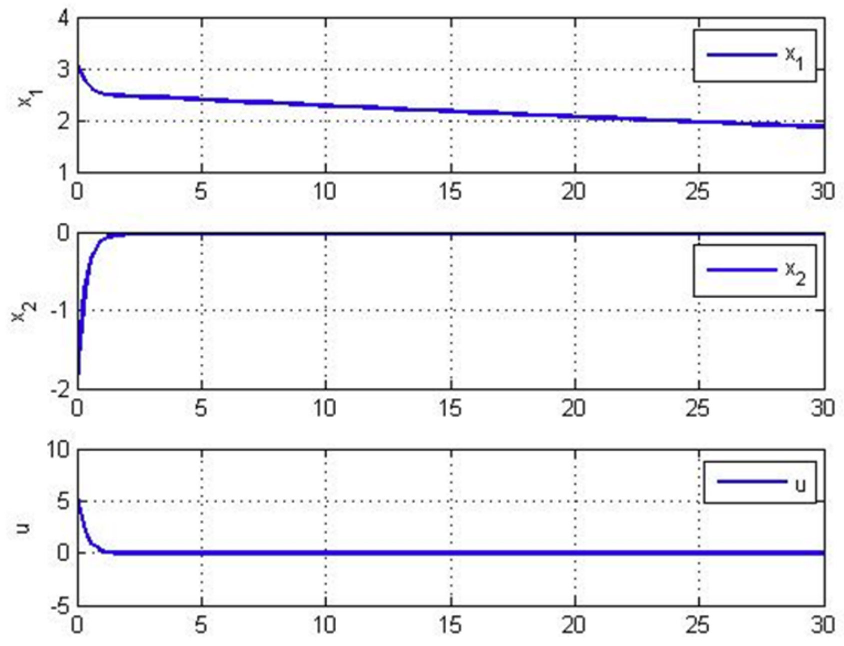
In this situation, the we would like to only penalize a non-velocity (we don’t care if the position of the block is non-zero). In order to influence the LQR algorithm to choose a controller which will behave in this fashion, we choose a *Q* matrix with a very small entry corresponding to position. One candidate to try is

$$Q = \begin{pmatrix} 0.001 & 0 \\ 0 & 10 \end{pmatrix} \qquad R = (1)$$

This yields

$$K = (\ 0.03 \quad 2.98\ )$$

The results of the system under this control law are shown below



As can be seen, we obtain the desired behavior. In this situation, the velocity of the block is quickly regulated to zero. However, since there is a very small penalty for a non-zero position, the controller does not exert much control authority to bring the position back to zero (x_1 at 30 seconds is still at approximately 1 unit).

<u>Summary</u>

Scenario	k_1	k_2	**Behaviour**
1	10	10.76	fast regulation of $\overline{x}(t) \to 0$, large $\overline{u}(t)$
2	0.03	0.12	slow regulation of $\overline{x}(t) \to 0$, small $\overline{u}(t$
3	0.03	2.98	only $x_2(t) \to 0$ quickly

LQR To Address FSFB Implementation Issues

So we see that LQR is a technique to allow us to design a FSFB controller that respects tradeoffs between states and controls. In our previous lecture (https://youtu.be/9vCTokJ5RQ8) we saw that while FSFB controllers had great power, there were some practical implementation issues to address, in particular:

1. Control saturation
2. Inability to measure the full state of the system

As Spiderman's uncle said, "With great power comes great responsibility"...

Example: DC Motor

In the previous lecture, we investigated some practical implementation issues with FSFB controllers using the DC motor as an example. Let us revisit this same problem using LQR control. Recall that the state and control vector for this system is

$$\overline{x}(t) = \begin{pmatrix} \theta(t) \\ \dot{\theta}(t) \\ i(t) \end{pmatrix} \qquad \overline{u}(t) = \begin{pmatrix} V_a(t) \\ T_L(t) \end{pmatrix}$$

The numerical state space representation we have for this system is

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -0.29 & 71.93 \\ 0 & -63.24 & -1020.35 \end{pmatrix} \qquad B_1 = \begin{pmatrix} 0 \\ 0 \\ 641.81 \end{pmatrix}$$

Again, we investigate the case where we only want to use V_a for control. Therefore, we have

$Q = 3\text{x}3$ weight matrix for the states
$R = 1\text{x}1$ weight matrix only penalizing V_a

Control Saturation

LQR is the perfect tool to address the control saturation issue. In the previous example we saw that if we were not careful with what we picked for closed loop poles, we might end up with a controller that demands hundreds of volts.

Joke: So in this case "with great power comes great electricity bills".
Peter Parker: "Ah gee uncle Ben, I didn't know my FSFB controller would be so aggressive"
Ben Parker: "Well Peter, just use LQR techniques to design a controller that is cognizant of control saturation issues"

If we hold Q constant and vary R, we can understand how the changes in R change the aggressiveness of the controller.

If we hold Q as shown below and then vary R we can investigate how this affects the resultant K

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad R = r_{11}$$

r_{11}	k_1	k_2	k_3
0.01	10	10.0021	8.65
0.1	3.16	3.1	2.05
1	1.0	0.93	0.34
10	0.32	0.25	0.05

As can be seen, by simply increasing the size of R relative to Q we can tune how aggressive the controller is. The larger the R the smaller the resultant control is.

<u>Inability to Measure Full State</u>

The problem of not being able to measure the full state of the system is still not directly addressable. Previously, we attempted to identify which eigenvalues were associated with which states. However due to the coupling we discussed in the previous lecture, an eigenvalue is typically associated with multiple states simultaneously. With LQR, we do not need to understand the relationship between eigenvalues and states, we can directly specify which states are important and which are not.

We can can now use the Q matrix to only penalize states that we can directly measure in the hope that this makes the the gains associated with the other states near zero. However, even with LQR this does not yield perfrect results. In this case we could try

$$Q = \begin{pmatrix} q_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad R = (1)$$

q_{11}	k_1	k_2	k_3	|	k_2/k_1	k_3/k_1
0.01	0.10	0.02	0.0014	|	0.193	0.014
0.1	0.32	0.05	0.0037	|	0.168	0.012
1	1.00	0.13	0.0092	|	0.131	0.009
10	3.16	0.29	0.02	|	0.090	0.006

As can be seen, this does seem to yield controllers where k_1 is the dominant term but there are still non-zero k_2 and k_3 terms. As such, a controller that implements only $V_a(t) = -k_1\, x_1(t)$ will still incur some penalties and performance degradations. This sets the stage for linear estimation where we will attempt to estimate the unmeasured states $x_2(t) = \omega(t)$ and $x_3(t) = i(t)$.

Summary

So in summary, LQR is a technique that can be used to generate a controller that is optimal in the sense that it solves the problem

$$(\wp)\ \underset{\overline{u}(t)\in\mathbb{R}^m}{\text{minimize}}\ J = \int_0^\infty \overline{x}(t)^T Q\,\overline{x}(t) + \overline{u}(t)^T R\,\overline{u}(t)\,dt \qquad \textbf{(Eq.2)}$$

$$\text{such that } \dot{\overline{x}}(t) = A\,\overline{x}(t) + B\,\overline{u}(t)$$

It turns out that the optimal control law is

$$\overline{u}(t) = -K\,\overline{x}(t)$$

where $K = R^{-1} B^T S$

Which we recognize as a full state feedback controller.

As a control engineer, we specify the Q and R matrices to directly influence the cost function and therefore the resulting solution. Q allows us to specify which states we care about driving to the origin and R allows us to specify how much of each control we are allowed to expend in order to achieve this return to the origin.