Control Bootcamp

Sondre Pedersen

February 14, 2025

1 Overview

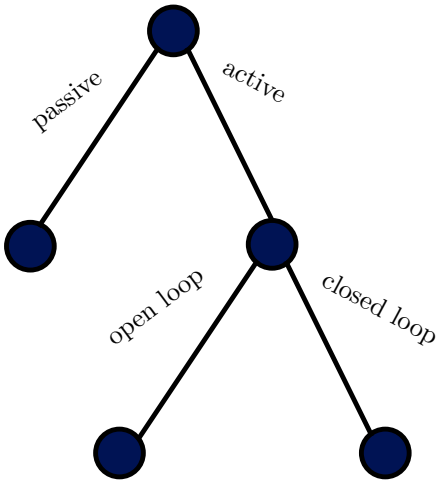
This document will quickly go through the highlights of optimal and modern control theory. This includes

- How to describe a control system with inputs and outputs in terms of a system of linear ODEs.
- How to design controllers to manipulate the behaviour of the system
- Design estimators (ex: Kalman filter) to reconstruct aspects of the system

This is just a high level overview, and not an exhaustive treatment of the material.

Some perspective - We can think of the world in terms of dynamical systems: systems of ordinary differential equations. This has been an extremely successful viewpoint for modelling real-world phenomenon. Fluid flow over a wing, population dynamics in a city, spread of a disease, the stockmarket, climate, and so on. Often we want to go beyond just describing the system, and manipulate/change the behaviour. This could be done by sending control inputs into the system in a pre-planned way, or you can measure the system and make decision based on what it is doing.

This is the overarching view in control theory: you describe some dynamical system of interest (pendulum, crane, et cetera) for which you design some control policy that changes the behaviour of the system to be more desirable.



Types of control - There are two main categories. (i) Passive control, where you make the control design ahead of time. An example of this is a streamlined tail section on a truck. These might make the air move around the truck in a more favourable way to reduce drag. If passive control is possible, it is usually a good idea. All the cost is up front. The flip side of this is (ii) active control. There are different versions of active control. The easiest version is open loop. Here you have a system with control inputs u and outputs y. Ahead of time, an open loop system will have to reverse engineer the behaviour you want, such that it can give the correct inputs.

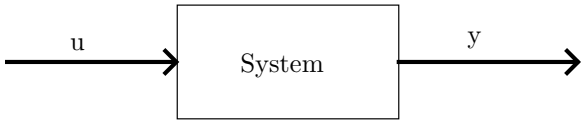


Figure 1: open loop system

An interesting example of this comes from balancing a pendulum. It is possible to rapidly move a pendulum up and down following a sinusoid pattern to balance it perfectly. The system here being the pendulum, and the input (u) being the motor moving it up and down. This control must be planned before starting the balancing, and cannot change should something go wrong.

To fix the issue of not being able to correct small errors, we design what is called closed loop feedback contol. These use sensors to keep track of how the system is behaving, and making correction based on that. This is the most interesting type of control mentioned so far.

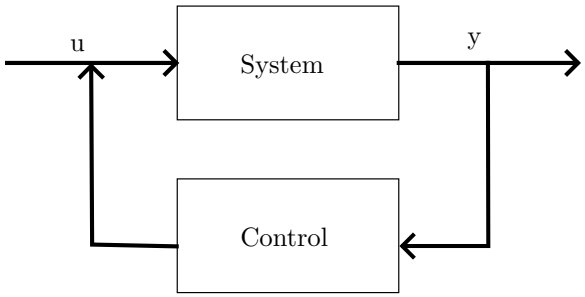


Figure 2: closed loop feedback

This feedback is useful to fix for inherent system uncertainty. Another big win for closed loop systems is that it can fix instabilities. For open loop systems, the system might still be unstable. This is different in closed loop, where we actually change the dynamics of the system, and can thus change it to be stable. The final point is that we can handle external disturbances if the controller is good enough.

Why can we change the stability? - In control theory we work with state space system of ODEs. This means that we have a state variable x, which is a vector containing all the relevant information for the system. For a pendulum this would typically be angle and angular velocity. A state-space ODE looks at the rate of change for the system.

$$\begin{aligned} \dot{x} &= Ax \\ \implies x(t) &= e^{At}x(0) \end{aligned}$$

The stability of this system depends on A. If any eigenvalues of A are positive, the system will be unstable. If all eigenvalues are negative, the system is stable. So in open loop control theory, we add another term which forces the system to become stable.

$$\dot{x} = Ax + Bu.$$

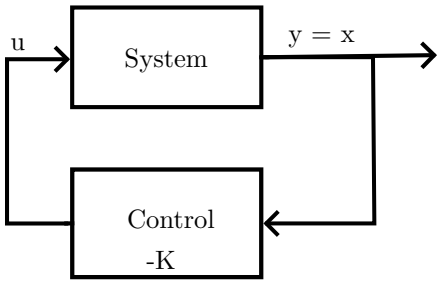
Here, u is our actuator, which allows us to manipulate the system. Another extension for later is the measurement of only certain aspects of the state.

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

This measurement might not include all the states if it is high dimentional. But if we assume that we can measure all of x. Then we can develop a control law $u = -Kx$ and put it back into the system equation:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ &= Ax - BKx \\ &= (A - BK)x \end{aligned}$$

This is now a new system, and with careful design we can force it to be stable.



2 Linear Systems

A brief review of linear systems. Again we are looking at $\dot{x} = Ax$ with the general solution $x(t) = e^{At}x(0)$. It is possible to take the Taylor expansion of e^{At}

$$e^{At} = I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 + \dots.$$

Directly computing this is not very practical. There are instead some tricks that allow us to do this in practice. We use the eigenvector and eigenvalues of A to get a coordinate transformation from the x coordinates into some eigenvector coordinates. In this coordinate system it is easier to write e^{At}, and also easier to understand the dynamics.

In general, and eigenvector or an eigenvalue satisfies the following properties:

$$A\xi = \lambda\xi.$$

The vector ξ here is very special. The relationship tells you that the transformation of ξ by A ($A\xi$) is just 'stretching' the vector. It is still pointing in the same direction as before the transformation. Writing down a matrix of all these special vectors $T = \begin{bmatrix}\xi_1 & \xi_2 & \dots & \xi_n\end{bmatrix}$ and the corresponding eigenvalues $D = \begin{bmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n\end{bmatrix}$ we can write $AT = TD$.

But why do we care about this relationship? Because we can use this T matrix to transform the state x into a more useful coordinate system:

$$\begin{aligned} x &= Tz \\ \dot{x} &= T\dot{z} = Ax \\ T\dot{z} &= ATz \\ \dot{z} &= T^{-1}ATz \\ \dot{z} &= Dz \qquad\qquad \text{using } AT = TD \implies T^{-1}AT = D \end{aligned}$$

This new system using z coordinates is much easier to work with, because the D matrix is a diagonal matrix (easy to work with). All the system components are now completely decoupled from each other. That means $\dot{z_1}$ is only a function of z_1 and so on.
The solution to the system (as before) is $z(t) = e^{Dt}z(0)$. The difference now is that e^{Dt} is easy to compute. Instead of the whole Taylor series expansion, we can write

$$z(t) = e^{Dt}z(0) = \begin{bmatrix}e^{\lambda_1 t} & & & \\ & e^{\lambda_2 t} & & \\ & & \ddots & \\ & & & e^{\lambda_n t}\end{bmatrix} z(0).$$

Going back to the original coordinate system is not hard either, by using $A = TDT^{-1}$ in the Taylor expansion.

$$\begin{aligned} e^{At} &= I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 + \dots \\ &= e^{TDT^{-1}t} \\ &= \mathcal{T}^{-1} + TDT^{-1}t + \frac{t^2}{2!}(TDT^{-1}TDT^{-1}) + \dots \\ &= T[I + Dt + \frac{1}{2!}D^2t^2 + \frac{1}{3!}D^3t^3 + \dots]T^{-1} \\ &= Te^{Dt}T^{-1} \end{aligned}$$

The simplificationis possible because all the terms in the middle cancel out: $T^{-1}T = I$. Plugging the result back into the original equation gives us

$$x(t) = Te^{Dt}T^{-1}x(0).$$

This has a nice interpretation:

- $T^{-1}x(0)$: transforming the initial conditions into initial conditions for the 'special' coordinate system $\implies z(0)$.
- $e^{Dt}T^{-1}x(0) \to e^{Dt}z(0)$ is advancing the initial condition to time t $\implies z(t)$.
- $Te^{Dt}T^{-1}x(0) \to Tz(t)$ is transforming the special coordinates at time t back into the original coordinate system.

3 Stability and Eigenvalues

Given a system, determine if it is stable or not. From before we have a solution to a linear system on the form

$$x(t) = Te^{Dt}T^{-1}.$$

The stability of the system depends on the e^{-1} term. From Euler's formula we know that $e^{\lambda t} = e^{at}(\cos(bt) + i\sin(bt))$, where $\lambda = a + ib$. The sum of cosine and sine will never be greater than 1, so the value of a alone determines if the value blows up to infinity over time. This will happen if a ¿ 0. When a ¡ 0, then the system will go to 0 over time.

Stability is slightly different in discrete time systems. We have individual measurements at discrete instances. So $x_{k+1} = \tilde{A}k_k$, where $x_k = x(k\Delta t)$. Also, $\tilde{A} = e^{A\Delta t}$.

Given an initial condition x_0, we can integrate the whole trajectory of the system, just by calculating the next step over and over.

$$\begin{aligned}
x_1 &= \tilde{A}x_0 \\
x_2 &= \tilde{A}x_1 = \tilde{A}^2x_0 \\
x_3 &= \tilde{A}^3x_0 \\
&\vdots \\
x_N &= \tilde{A}^Nx_0
\end{aligned}$$

Since $\tilde{A}$ is a function of the diagonal matrix D, the effect of squaring $\tilde{A}$ is to multiply the λ. To really understand this, some more complex analysis would be necessary. The important point is just that the eigenvalues of $\tilde{A}$ must have magnitude less than 1 to be stable. A single eigenvalue with magnitude greater than 1 would cause the system to be unstable.

4 Linearizing around a fixed point

If we have a non-linear system $\dot{x} = f(x)$ it is still possible to get it on the form $\dot{x} = Ax$ with linearization around a fixed point.
The basic steps are:

- Find fixed points $\overline{x}$ s.t. $f(\overline{x}) = 0$
- Linearize about $\overline{x}$ by finding the Jacobian at $\overline{x}$: $\frac{\partial f}{\partial x}$

$$\begin{aligned}
\dot{x} &= f(x) \\
&= f(\overline{x}) + \frac{\partial f}{\partial x}\Big|_{\overline{x}} \times (x - \overline{x}) + \frac{\partial^2 f}{\partial x^2} \times (x - \overline{x})^2 + \dots
\end{aligned}$$

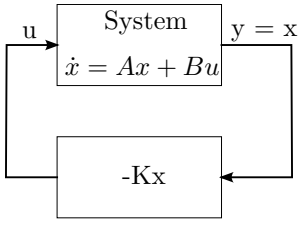
If we evaluate points very close to the fixed points we get a close approximation $\Delta\dot{x} = \frac{\partial f}{\partial x}\big|_{\overline{x}} \times \Delta x \implies \Delta\overline{x} = A\Delta x$
So the point is that by zooming really close into the fixed point using the Jacobian, we get linear dynamics.
There are more details here but I am skipping over it for now

5 Controllability

We are finally going to add control inputs in order to bend it to our will. This is done by adding a term to the system equations:

$$\dot{x} = Ax + Bu.$$

where u is the control input signals. This will allow us to shape the eigenvalues of the system. For now we will pretend that we can measure the whole state of the system (y = x), and feed this measurement into the controller.



So now we are measuring the system. If it seems to move away from where we want it to be, the controller will kick in and force the system to remain where we want it to be. For a controllable system, $u = -Kx$ is the best for linear systems. With this feedback control, we can put it back into the original equation

$$\begin{aligned}\dot{x} &= Ax + Bu \\ &= Ax - Bkx \\ &= (A - BK)x\end{aligned}$$

We did this before. But the point is that we can choose K to make the system behave as we want (get the eigenvalues that we want). This is possible for systems that are completely controllable. Additionally, it might be possible to steer the system anywhere we want, with the right inputs of u (as long as the linear approximation is valid).

Controllability is loosely: when you can choose $u = -Kx$ and place the eigenvalues of the system anywhere we want. In real life systems (as a control engineer) you typically don't get to choose A. It is given to you (from the wings of an airplane for example) and it doesn't behave as expected. You typically also don't get to choose B (the plane only has certain sensors / information). So the only thing we have any control over is u.
Given A and B, it is possible to determine right away if the system is going to be easy to control, or maybe even impossible. So figuring this out could save a lot of time.

An example of a controllable system is

$$\begin{bmatrix}\dot{x_1} \\ \dot{x_2}\end{bmatrix} = \begin{bmatrix}1 & 0 \\ 0 & 2\end{bmatrix}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} + \begin{bmatrix}0 \\ 1\end{bmatrix} u.$$

By looking at this equation, we can see that it cannot be controllable. The B matrix ($\begin{bmatrix}0 & 1\end{bmatrix}^\top$) leaves no option on how to change $\dot{x_1}$. $\dot{x_2}$ is controllable, but that does not help. An obvious example of a controllable system would be

$$\dot{x} = \begin{bmatrix}1 & 0 \\ 0 & 2\end{bmatrix} + \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}u_1 \\ u_2\end{bmatrix}.$$

This is easy to control because we have two actuators (two columns in the B matrix), one for each of the output values. The following system is also controllable, but it is not as obvious

$$\begin{bmatrix}1 & 1 \\ 0 & 2\end{bmatrix}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} + \begin{bmatrix}0 \\ 1\end{bmatrix} u.$$

Here we have only one actuator, but since the system is coupled, you can still provide a value for u and get the behaviour that you want. It is much harder to pick the correct u here though, so more math is needed. The advantage of this is that you can get away with fewer sensors and acutators, and still get the behaviour that you want.
To determine the controllability of a system, we look at this matrix:

$$\mathcal{C} = \begin{bmatrix}B & AB & A^2B & \cdots & A^{n-1}B\end{bmatrix}.$$

and check if it has full column rank. In an n-dimensional system, if $\mathcal{C}$ has n linearly independent columns, then the system is controllable. If the matrix is rank deficient, the system is uncontrollable (there are some directions in $\mathcal{R}^n$ that we simply cannot touch).

Equivalences -
The following three things are the same (important equivalences)

- The system is controllable
- Arbitrary eigenvalue (pole) placement. $u = -Kx \implies \dot{x} = (A - BK)x$
- Reachability (full in $\mathbb{R}^n$). Reacheble set $R_t = \{\xi \in \mathcal{R}^n |$ there is an input u(t) so that $x(t) = \xi\}$. So reachability means that $R_t = \mathbb{R}^n$.

Discrete-time impulse response -
To understand more about how the controllability matrix works, we look at the system in discrete time

$$x_{k+1} = Ax_k + Bu_k.$$

Now give a short kick to the system with u (impulse). $u_0 = 1,\ u_1 = u_i = 0$. How does this affect the system? $x_0 = 0,\ x_1 = B,\ x_1 = AB,\ x_2 = A^2B, \ldots, x_m = A^{m-1}B$. So each timestep the system takes is another column in the controllability matrix, when the system starts with a kick. Interesting, but does this mean anything? This matrix is a representation of anything that can be reached by control, so if the matrix does not span $\mathbb{R}^n$, then those states cannot be reached.

Gramians - The solution to the system $\dot{x} = Ax + Bx$ is

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

The integral part of this function is the convolution between e^{At} and $Bu(t)$. e^{At} works like a kernel, sliding accross the control input. But more on this later. The point is that we have something called the controllability Gramian that looks like

$$W_t = \int_0^t A^{A\tau}BB^\top e^{A^\top \tau}d\tau.$$

This mathematical object lives in $\mathbb{R}^{n\times n}$. If we find the eigenvalue decomposition of this object $W_t\xi = \lambda\xi$ we can order the eigenvalues from biggest to smallest, it gives us information about which directions are the most controllable. In other words, the directions it makes most sense to "spend energy".

In discrete time systems, we can approximate it with

$$W_t \approx \mathcal{C}\mathcal{C}^\top.$$

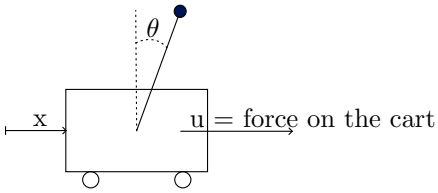
Another way to find it is to use a singular value decomposition of the controllability matrix ($[U, \Sigma, V] = svd(\mathcal{C})$). Here, the eigenvalues of the Gramian equal the values of Σ squared. The point is that this will also give you an ordered list of "the best directions".

Other test -
Other tests for controllability includes PBH. It says that (A, B) is controllable iff

$$rank\left(\begin{bmatrix} A - \lambda I & B \end{bmatrix}\right) = n \qquad \forall \lambda \in \mathcal{C}.$$

6 Inverted Pendulum on a Cart

The classic pendulum example, but now on a cart. The goal is to control the cart to keep it upright.



We have the state, $\mathbf{x}$, and the state derivative $\dot{\mathbf{x}}$

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}, \ \dot{\mathbf{x}} = f(\mathbf{x}).$$

This is a nonlinear problem / state derivative. It comes from physics and will not be derived here. We are able to linearize the system. Intuitively we know that the system has two fixed points: pendulum in the up or down position.

These are when $\theta = 0, \pi$, $\dot{\theta} = 0$, and $\dot{x} = 0$. The position is irrelevant (a free variable) for the fixed points. Deriving the state derivative for this system is actually a bit nasty, so I am not going to write it down here. The point is that the control inputs show up in $\dot{x}$ and $\dot{\theta}$ expressions.
Steps to make a controller:

- Linearize around the fixed point (upright position) to uptain A and B
- Check that the rank of the controllability matrix is 4
- Decide what eigenvalues you want
- Create K (K = place(A, B, eigenvalues) in matlab)

With the K gain matrix, we have everything needed to stabilize the system. First we need some reference values that we would like the system to reach. An example could be $\begin{bmatrix} 1 & 0 & \pi & 0 \end{bmatrix}^\top$. The choice of eigenvalues determine how fast this goal state will be reached. Very negative values will cause rapid change, but will also require high control inputs. In a real system that might put too much torque on wheels or similar, so faster is not always better. Additionally, eigenvalues that are too extreme might cause the system to become unstable again, even if they are all negative.

7 Linear Quadratic Regulator (LQR)

The linear quadratic regulator tries to find the best eigenvalues. Instead of the control engineer having to try different values over and over, this method will find them directly.

The principle behind this method is that we put a cost on a lightly damped system (one that takes a long time to reach goal state), along with a cost on high control inputs. This cost function looks like this

$$J = \int_0^\infty x^\top Q x + u^\top R u \; dt.$$

Here we can interpret Q to be a matrix determining how 'bad' it is when x is not where we tell it to be, and R to determine how 'bad' it is to have high control inputs. The integration will then add up the penalties over time.

- $x^\top Q x$ must stabilize quickly for this to be small
- $u^\top R u$ must spend little energy doing so

So the goal is to find eigenvalues such that these two quantities are as small as possible. An example could be

$$Q = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 10 & \\ & & & 100 \end{bmatrix}.$$

This is telling us that we penalize being in the wrong position and having wrong velocity low (1 unit). But having the wrong angle is penalized more harshly (10 units) and wrong angular velocity even harsher still (100). An example of R could be 0.001 (here we only have one control input, so R is a 1x1 matrix). This might be reasonable if our motor on the cart can handle high actuation.

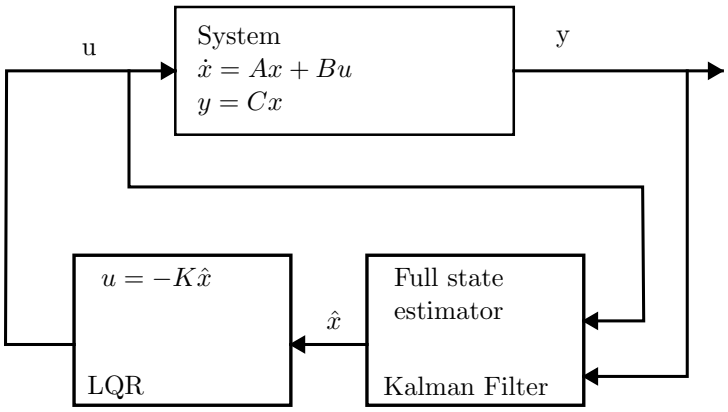
So if we have a Q and R, there exist an optimal K matrix that minimizes the cost function. That is the linear quadratic regulator. Linear, because it is a linear full state feedback controller. Quadratic because it minimizes a quadratic cost function. Regulator, because it will stabilize the system. The calculations can be challenging, but can be done using optimization theory. It involves the Riccati equation, which is expensive for high-dimentional systems. But since these calculations have been solved by other smart people, it is not the hardest part of making a control system. That would usually be making the correct model.

8 Full-State Estimation

Until now we have assumed that we can measure the system perfectly. But in real life, we often do not have access to the full state of the system. To use the pendulum example, maybe we only have measurements on the angle. The question then becomes, can we reconstruct the rest of the state, based on this one measurement? The equations describing our system now becomes

$$\dot{x} = Ax + Bu \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^q$$
$$y = Cx \qquad y \in \mathbb{R}^p$$

Now we will be looking at the observability of A and C. We ask if we can estimate any state x from measurements y. Appending this estimator to the schematic we get

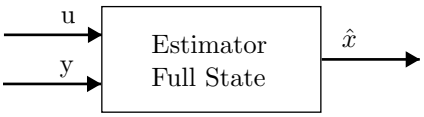


The interesting thing is that the same techniques used to get LQR show up again when trying to get the optimal Kalman filter. There is some dualtiy between controllability and observability. Even more interestingly, it is possible to develop the optimal controller and estimator separately. When put together, the combination will also be optimal.

Observability -
We have a similar observability matrix as we had for controllability. It looks like

$$\mathcal{O} = \begin{bmatrix} C & CA & CA^2 & \cdots & CA^{n-1} \end{bmatrix}^\top.$$

The Kalman filter is going to be itfes own linear dynamical system, and its eigenvalues will tell us how fast $\hat{x}$ converges to x. Before we get to the Kalman filter, we take a closer look at estimation. I cannot call this estimator a Kalman filter, because it will not be optimal (yet).



$$\frac{d}{dt}\hat{x} = A\hat{x} = Bu + K_f(y - \hat{y})$$
$$\hat{y} = C\hat{x}$$

This system looks a lot like the system for x. The difference is that we have the $K_f(y - \hat{y})$ term. It will take the difference between a new measurement and what we think that measurement should be. This difference will help update the state estimation. Putting the two equations together we get

$$\begin{aligned} \frac{d}{dt}\hat{x} &= A\hat{x} + Bu + K_f y - K_f C\hat{x}k \\ &= (A - K_f C)\hat{x} + \begin{bmatrix} B & K_f \end{bmatrix} \begin{bmatrix} u \\ y \end{bmatrix} \end{aligned}$$

By looking at the error, it can be found that this estimator can get as close as you want to measuring the real state. In reality however, there will be some sensor noise and disturbance to the system, making an aggressive estimator sub-optimal. The Kalman Filter solves this problem.

Adding noise and disturbance - Let disturbance w_d and sensor noise w_n follow a gaussian distribution. The system now becomes

$$\dot{x} = Ax + Bu + w_d$$
$$y = Cx + w_n$$

To find the best eigenvalues for the full state estimator, minimize the following cost function:

$$J = \mathbb{E}((x - \hat{x})^\top (x - \hat{x})).$$

It looks complicated (and is), but it follows the same math as finding the optimal K_r from LQR.