Chapter 1

Introduction

Without control systems there could be no manufacturing, no vehicles, no computers, no regulated environment—in short, no technology. Control systems are what make machines, in the broadest sense of the term, function as intended. This course is about the modeling of physical systems and the analysis and design of control systems.

1.1 Prologue

This is your first course on control. It may be your one and only course on control, and therefore the aim is to give both some breadth and some depth. Mathematical models of physical systems is an important component. There's also an introduction to state models. Finally, there's an introduction to design in the frequency domain.

The sequels to this course are ECE488 Multivariable Control Systems, which develops the state-space method in continuous time, and ECE481 Digital Control Systems, which treats digital control.

There are several computer applications for solving numerical problems in this course. The most widely used is MATLAB, but it's expensive. A popular free alternative is Scilab. Others are Mathematica (expensive) and Octave (free).

From SE380 it is hoped that you learn the following:

- The value of block diagrams.
- What feedback is and why it's important.
- How to model a system using differential equations.

- What it means for a system to be linear.
- The value of graphical simulation tools like Simulink.
- Why we use transfer functions and the frequency domain.
- What stability means.
- How to determine if a feedback system is stable by checking poles.
- The Nyquist stability criterion and the meaning of stability margin.
- How to design a simple feedback loop using frequency-domain methods.
- What makes a system easy or hard to control.

1.2 Familiar Examples

We begin with some examples that everyone knows. A familiar example of a control system is the one we all have that regulates our internal body temperature at 37° C. As we move from a warm room to the cold outdoors, our body temperature is maintained. Nature has many other interesting examples of control systems: Large numbers of fish move in schools, their formations controlled by local interactions; the population of a species, which varies over time, is a result of many dynamic interactions with predators and food supplies. And human organizations are controlled by regulations and regulatory agencies.

Other familiar examples of control systems:

- autofocus mechanism in cameras
- cruise control system in cars
- thermostat temperature control systems.
- auto-pilot on a plane

One of the things you will notice during the course is that we analyze control systems in a general way. This means that we will develop tools that are widely applicable and can be used in a lot of different application areas. These include

- flow control in data networks
- bioengineering applications
- economic modeling and policy development
- operating systems [1]
- Middleware (application servers, database management systems, email servers) [1].

In this course we will mostly use mechanical examples since their behaviour is easier to understand than, say, electromagnetic systems, because of our experience in the natural world. When possible we will try to incorporate examples where control is used in computing systems.

1.3 What is control engineering?

In control engineering, the 'system' to be controlled is termed the plant. Control engineering deals with changing the behaviour of the plant in a desirable manner in the presence of modeling uncertainty and despite uncontrolled influences (disturbances).

We typically change the behaviour of the plant by connecting it in a feedback loop to another 'system' called the the controller, or, sometimes, the compensator.

For example, in helicopter flight control, the plant is the helicopter itself plus its sensors and actuators. The controller is implemented as a program running in an on-board computer. The disturbance might be the force of the wind acting on aircraft. The design of a flight control system for a helicopter requires first the development of a mathematical model of the helicopter dynamics. This is a very advanced subject, well beyond the scope of this course. We must content ourselves with much simpler plants.

Figure 1.1 gives an abstract representation of a general feedback control system. Figure 1.2 provides a more concrete example of the components in a control system. This is a cruise control system for a car; a classical example.

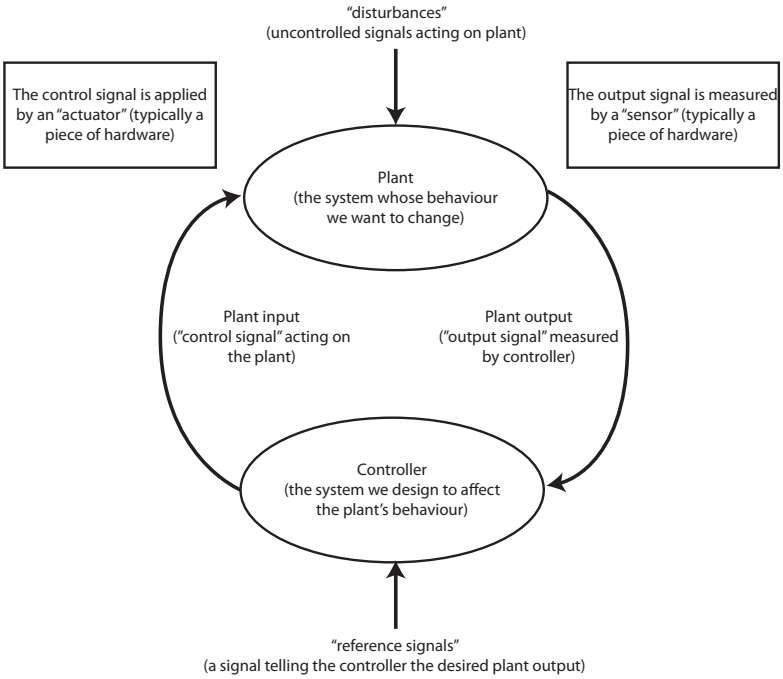

Figure 1.1: Conceptual drawing of a general control loop.

1.4 Control engineering design cycle

The typical control engineering design cycle involves the following steps and is illustrated in Figure 1.3.

1. Get problem specification. Typical specifications include: closed-loop stability, "good" steady-state tracking, disturbance rejection, transient specifications. These specifications are typically expressed in time and frequency domains and we will learn to move between them with ease.

2. Model the plant. By "model" we mean obtain a mathematical model of the plant. Typically the model will be one or more differential equations. Experiments may be needed to obtain numerical values of some parameters ("system identification").

3. Obtain a transfer function for the plant. Classical control, which is the focus of this course, requires that we have a transfer function for the plant. Linear time-invariant (LTI) systems, and only LTI systems,

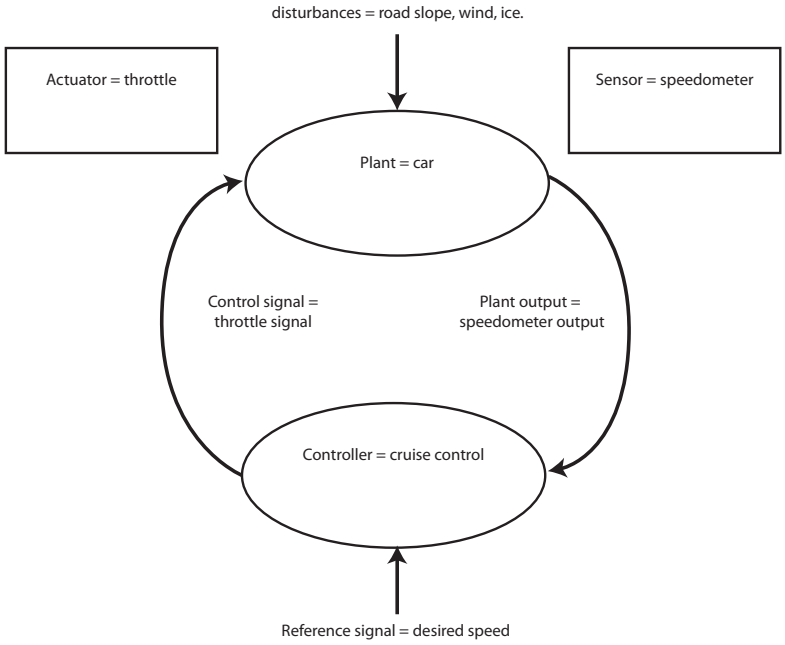

Figure 1.2: Components in a cruise control system.

have transfer functions. The transfer function (TF) of an LTI system is defined to be the ratio of the Laplace transform of the output to the Laplace transform of the input where the LTs are taken with zero initial conditions[1].

4. Design a controller. The controller itself is a transfer function. It corresponds to an ordinary differential equation (ODE) that relates the controller outputs (the control signal) to the controller inputs.

5. Simulation. It is important to simulate how the controller and plant work together since approximations were likely made along the way. Computer simulation is faster and cheaper compared to testing on the the controller on the real plant.

6. Implement the controller. You can build a system whose transfer function matches that designed in step 4. More realistically, the controller is implemented on a computer as a difference equation.

[1] Note that if the plant is nonlinear, which virtually every system is, then we will need to first linearize it before we can obtain a transfer function.

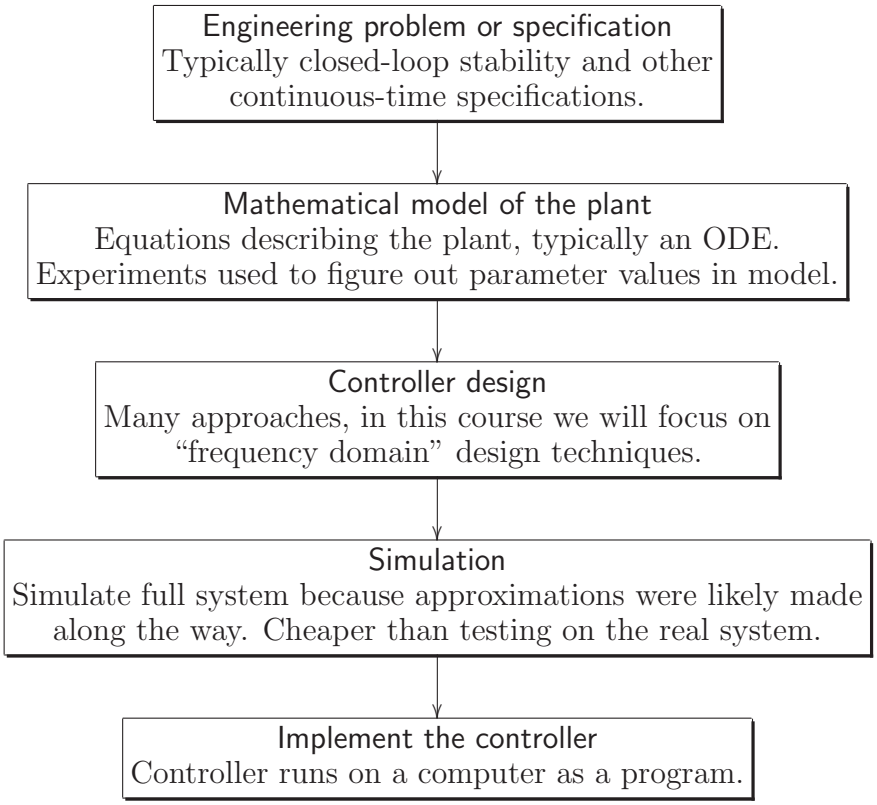

Figure 1.3: The control system design process

1.5 Basic unity feedback control system

Instead of using the abstract drawing of a control system in Figure 1.1 we use block diagrams to represent control systems. We will formally study block diagrams later on. In this section we introduce the block diagram of one of the most common control systems : the unity feedback continuous-time control system. This block diagram is shown in Figure 1.4. Henceforth we will almost exclusively deal with this system.

In this diagram the plant model $P(s)$ is a transfer function derived using physical laws and / or system identification techniques. We will review how to obtain transfer functions for various systems in this course. The symbol $Y(s)$ denotes the Laplace transform of the system output signal $y(t)$. The symbol $U(s)$ denotes the Laplace transform of the system input signal $u(t)$.

The controller $C(s)$ is also a transfer function. It is designed by the

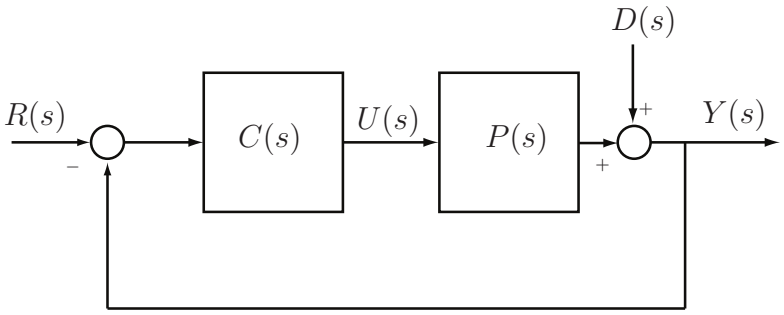

Figure 1.4: Continuous-time unity feedback control system.

control engineer. We will learn to design controllers in this course. Finally, $R(s)$ is the Laplace transform of the reference signal $r(t)$ and $D(s)$ denotes the Laplace transform of the disturbance signal $d(t)$. Typical control objectives include

- closed-loop stability.
- Tracking a step or ramp input signal, i.e., $y(t) \to r(t)$ as $t \to \infty$ for a family of reference signals.
- Adequate transient performance and robustness to model uncertainty and disturbances.

In this course we will understand theses specification in great detail. We develop techniques for designing control laws that change the behaviour of the given system to ensure that these objectives are met.

There are various assumptions that we are implicitly making when we draw a system in this form.

- The controller and plant are linear and time-invariant.
- The controller has access to the error signal $e := r - y$ and not the output itself. It is possible to make a more general scheme where the controller has access to r and y separately, but we'll stick to this simpler scheme.
- The disturbance is added at the output of the plant. Another common configuration has the disturbance being added at the input.

- The sensor is perfect since there is no transfer function in the feedback path.
- The actuator is perfect or, at the very least, has been incorporated into the plant model $P(s)$.

The typical control objectives we are interested in are

- Tracking: we want to y to track r.
- Disturbance rejection: we want to ensure that d does not significantly affect y.
- Robustness: we want "good" tracking and disturbance reaction despite uncertainty in the plant model $P(s)$.

1.6 Control without feedback

Before analyzing the unity feedback architecture, we first look at simpler control scheme that does not involve feedback. Consider the basic block diagram for the feedforward continuous-time control system or open-loop continuous-time control system shown in Figure 1.5. This architecture is appealing because

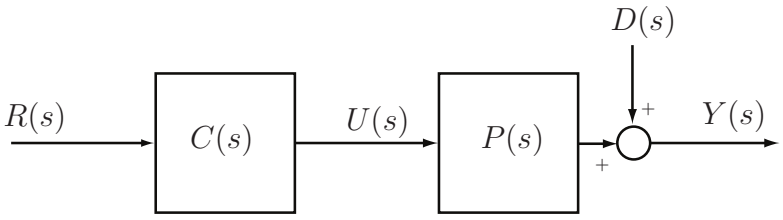

Figure 1.5: Feedforward or open-loop control system.

it is very simple and it is easy to implement since we don't need a sensor for feedback. However, we will see that this architecture makes designing a control law $C(s)$ that achieves the control objectives described in the last section impossible.

Let's analyze the open-loop architecture in the context of tracking, disturbance rejection and robustness

(i) Tracking. If we ignore d for the moment, i.e., set $d = 0$, then for perfect tracking we want $y = r$. This suggests the control law

$$C(s) = \frac{1}{P(s)}.$$

With this control law the transfer function from $R(s)$ to $Y(s)$ is

$$\frac{Y(s)}{R(s)} = 1$$

and we have perfect tracking for any reference input.

(ii) Disturbance rejection. Ignoring r for the moment, i.e., set $r = 0$, for ideal disturbance rejection the disturbance would have no affect on the output. In other words, for ideal disturbance rejection we want the transfer function from $D(s)$ to $Y(s)$ to be

$$\frac{Y(s)}{D(s)} = 0.$$

However, in the open-loop setup we have that $y = d$ so that the transfer function from $D(s)$ to $Y(s)$ is

$$\frac{Y(s)}{D(s)} = 1.$$

In other words, there is no disturbance rejection at all.

(iii) Robustness. A typical way to measure this is to look at the sensitivity of the transfer function from $R(s)$ to $Y(s)$. Denote by $T(s)$ the transfer function from $R(s)$ to $Y(s)$

$$T(s) := \frac{Y(s)}{R(s)} = C(s)P(s).$$

We want to characterize the relative perturbation in T due to a relative perturbation in P

$$\begin{aligned} \lim_{\Delta P \to 0} \frac{\Delta T/T}{\Delta P/P} &= \lim_{\Delta P \to 0} \frac{\Delta T}{\Delta P} \frac{P}{T} \\ &= \frac{\mathrm{d}T}{\mathrm{d}P} \cdot \frac{P}{T} =: S(s). \end{aligned}$$

So S is a measure of the sensitivity of the transfer function to variations in the plant transfer function

$$S(s) = \frac{\mathrm{d}T}{\mathrm{d}P} \cdot \frac{P}{T}. \tag{1.1}$$

Ideally the sensitivity would be zero indicating that perturbations in the plant have no affect on the transfer function. In this case we have

$$S(s) = \frac{\mathrm{d}T}{\mathrm{d}P} \cdot \frac{P}{T} = C(s)\frac{1}{C(s)} = 1.$$

Hence the sensitivity is always 100%.

In summary, the open-loop architecture has the following traits

- Tracking can be made theoretically perfect.
- disturbance rejection is poor.
- poor robustness.

Actually, there are two subtle problems with even tracking for open-loop systems. First, suppose that $P(s)$ is a rational, strictly proper transfer function[2]. This is very typical for many plants. Then the control law that provides perfect tracking

$$C(s) = \frac{1}{P(s)}$$

is improper. This is a problem because improper transfer functions are non-causal and hence can only be approximately implemented. For example, if $P(s) = 1/(s+1)$ then $C(s) = s + 1$. The term s in $C(s)$ corresponds to a pure differentiator and can only be approximated.

A more serious problem occurs if $P(s)$ has any poles or zeros in the closed right-half complex plane (CRHP)

$$\{s \in \mathbb{C} : \mathrm{Re}\,(s) \geq 0\}.$$

[2]A transfer function $P(s) = \frac{N(s)}{D(s)}$ is rational if $N(s)$ and $D(s)$ are both polynomials. It is proper if the degree of the numerator is less than or equal to the degree of the denominator, i.e., $\deg(N(s)) \leq \deg(D(s))$. It is strictly proper if $\deg(N(s)) < \deg(D(s))$.

The feedforward controller $C(s) = 1/P(s)$ cancels out all the poles and zeros of the plant $P(s)$. If $P(s)$ has any poles or zeros in the CRHP then this controller will not work. For example, let

$$P(s) = \frac{s-1}{s-2}, \qquad C(s) = \frac{s-2}{s-1}.$$

Canceling the pole at $s = 2$ yields the following problems

- Any noise at the plant controller interface, and there will always be noise at every possible point in a control system, must pass through an unstable system $P(s)$ before showing up at y. Hence the output y will "blow up" even if $r = 0$.
- Any non-zero initial conditions on the plant will also make the plant output blow up.

Canceling the zero at $s = 1$ means that the control signal u will become unbounded for most reference signals. Since most actuators cannot handle arbitrarily large control signals we will get saturation of our control signal.

1.7 Control with feedback

Consider once again the basic unity feedback control system in Figure 1.6. We now show that feedback can overcome the deficiencies we've observed in

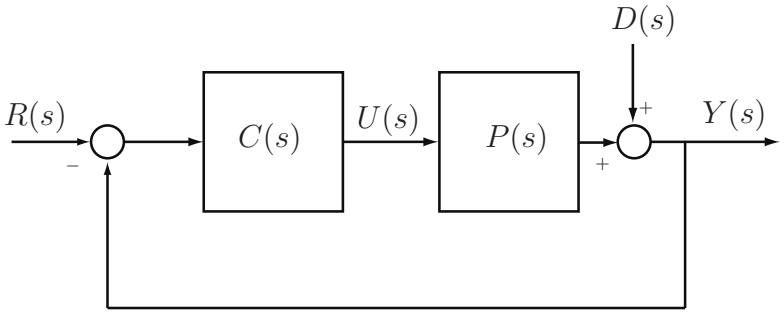

Figure 1.6: Continuous-time unity feedback control system.

open-loop control. Let's analyze the closed-loop architecture in the context of tracking, disturbance rejection and robustness

(i) Tracking. If we ignore d for the moment, i.e., set $d = 0$, we have that the transfer function from $R(s)$ to $Y(s)$ is

$$\frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)}.$$

Ideally this transfer function will equal one. Thus it seems that we can get good tracking by choosing "high gain control", i.e.,

$$C(s) = K$$

with K being a large constant gain.

(ii) Disturbance rejection. Ignoring r for the moment, i.e., set $r = 0$, we get that the transfer function from $D(s)$ to $Y(s)$ is

$$\frac{Y(s)}{D(s)} = \frac{1}{1 + C(s)P(s)}.$$

Ideally, we'd like this transfer function to be close to zero. Once again "high-gain control" seems like it will work.

(iii) Robustness. We once again check the sensitivity of the transfer function from $R(s)$ to $Y(s)$

$$T(s) = \frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)}$$

with respect to perturbations in the plant $P(s)$. As before, the sensitivity of the transfer function $T(s)$ to the plant $P(s)$ is given by (1.1) and re-written here

$$S(s) = \frac{\mathrm{d}T}{\mathrm{d}P} \cdot \frac{P}{T}.$$

In this case we have

$$S(s) = \frac{\mathrm{d}T}{\mathrm{d}P} \cdot \frac{P}{T} = \frac{C(s)}{(1 + C(s)P(s))^2} \frac{1 + C(s)P(s)}{C(s)} = \frac{1}{1 + C(s)P(s)}.$$

Since we want small sensitivity, high gain control seems to do the job.

The above analysis suggests that we have solved our control design problem by simply using high gain control. Unfortunately things are not as simple as they appear.

- In the presence of any time delay, the system will always go unstable for high K.
- Even if there is no delay (which is not realistic), the closed-loop system will usually go unstable for high K.
- High gain control can result in large control signals that can saturate actuators.

In summary, feedback control has the potential to meet our goals. Although a simple high gain controller $C(s) = K$ will rarely work in practice, the concept of high gain control is important. We'll see that a good control strategy is to use high gain in well-chosen frequency ranges.

1.8 Computing systems example

In this section we present an example of a computing system in which feedback control can be applied. This example is taken from [1] which is available online through the University of Waterloo library website. The reference [1] contains other examples that the interested reader can consult.

A central element of the Internet is TCP, the transmission control protocol, which provides end-to-end communication across network nodes. The designers of TCP were concerned about regulating traffic flows in the presence of network congestion. One way in which this regulation occurs is at routers that direct packets between endpoints. In particular, routers have finite-size buffers. Thus, to prevent buffer overflows during congestion, routers may discard packets (which results in their later retransmission as part of the TCP protocol).

Unfortunately, by the time that buffer overflows occur, it may be that the network is already congested. The idea behind random early detection (RED) is to take action before congestion becomes severe.

Random early detection measures how much of critical buffers are consumed, a metric that is referred to as the buffer fill level (BFL). As depicted in Figure 1.7, RED introduces the capability of randomly dropping packets even if buffer capacity is not exceeded. If BFL is small, no packets are dropped. However, as BFL grows larger, progressively more packets are dropped.

Random early detection has been shown to reduce network congestion and to improve network throughput. However, one challenge with using RED

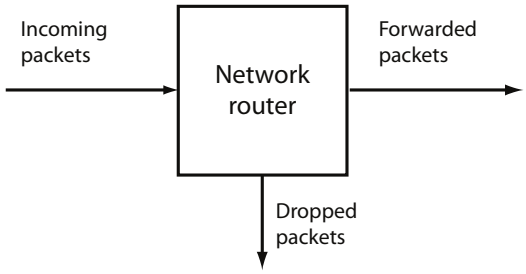

Figure 1.7: Router operation under random early detection. Incoming packets are dropped based on an adjustable drop probability. Drop probability controls the buffer fill level inside the router.

in practice is specifying its configuration parameters, especially the BFL at which packets start being dropped and the maximum drop probability. One approach to this tuning has been to view RED as a feedback control system with a regulation objective in which the reference input is a desired BFL and the control input is drop probability. Figure 1.8 depicts this perspective. Control theory is used to study the impact on stability and other properties for different settings of RED configuration parameters. Proportionalintegral (PI) control has been used for management of the accept queues of network routers.

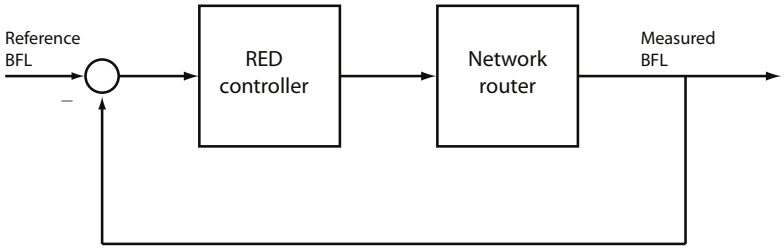

Figure 1.8: Feedback control of buffer fill level in a router. The reference input is the desired BFL, and the drop probability is the control input.

1.9 Very brief history

1. Ancient control (pre-1940): ad hoc control, almost no analysis tools. Examples: water clock (2^{nd} century BCE). More recently, in 1769 a

feedback control system was invented by James Watt: the flyball governor for speed control of a steam engine. Stability of this control system was studied by James Clerk Maxwell, whose equations you know and love.

2. Classical control (1940's-1950's) : this constitutes the main topics covered in this course. Deals mostly with single-input single-output (SISO) linear systems. Heavy use of "frequency domain" interpretation of linear systems including Bode plots and Nyquist methods. Specifications are based on closed-loop gain, bandwidth, and stability margin. Design is done using Bode plots. Controllers are typically tuned by hand. Examples: anti-aircraft guns (WWII), chemical plants, nuclear reactors, cruise control in cars, motor control.

3. Modern control (1960's - 1970's) : The approach to control analysis and design, which is the subject of ECE488, is in the time domain and uses state-space models instead of transfer functions. Specifications may be based on closed-loop eigenvalues, that is, closed-loop poles. This approach is known as the "state-space approach".

4. "Post modern" control (1980's - present) : frequency domain tools and thinking but state-space computational methods; robust control; optimal control (where the controller is designed by minimizing a mathematical function). In this context classical control extends to $\mathcal{H}^\infty$ optimization and state-space control extends to Linear-quadratic Gaussian (LQG) control. Discrete-event systems (DES); Hybrid systems. Examples: advanced aircrafts, unmanned vehicles, large power systems, medical applications, control over networks, distributed control.