Recitations

Recitation 1
Discrete and continuous systems

The goals of this chapter are:

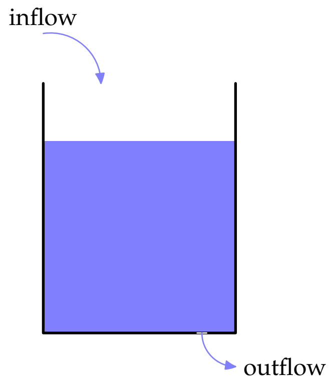
- to turn a description of a first-order, continuous-time system into a differential equation;
- to understand the system's behavior using a discrete-time approximation suitable for paper-and-pencil calculations; and
- to formalize this approximation into the general-purpose, forward-Euler method for numerically solving differential equations.

This chapter introduces a continuous-time system – the leaky tank – and its discrete-time approximation. Continuous- and discrete-time systems are often related because of how physics in general and computers in particular work. In physics, time is a continuous variable. Yet computers are discrete-time machines; more accurately, they are continuous-time devices whose behavior is accurately approximated in discrete time. Computers are designed in this way partly to minimize synchronization problems. Therefore, when a computer simulates a physical system – as opposed to solving analytically the equations that describe the system – you have to approximate the continuous-time system using the discrete-time behavior of the computer.

The leaky tank is perhaps the simplest interesting continuous-time systems and, as we find in later chapters, is a building block: an element from which to build complex systems. Similarly, its discrete-time approximation is the simplest interesting discrete-time system and is a building block for complex discrete-time systems. In this chapter we translate the leaky-tank model into continuous-time mathematics, use discrete-time approximations to understand its behavior qualitatively and quantitatively, and use the model to see why the water at the seashore is warmer in August than it is in June.

1.1 Leaky tank

Water flows into and leaves the leaky tank. We would like to describe the system mathematically. Then we can answer questions like 'What happens if you dump a bit of water in: How fast does the water leak out and how does that rate change with time?' This question may seem dull – what a simple input! – but keep your eyes on the prize: to understand this simple system because it is a building block for complex systems. One way to understand a system is to **play** with it: to try many inputs and to look at their outputs. So we play with it. As a precursor to trying various inputs, we need a mathematical description or model.



1.1.1 Mathematical model

To describe the system mathematically, we need a model of water leakage. The simplest model is that water leaks at a constant rate, independent of the height in the tank. That model is, however, too simple to show interesting behavior. So try the next-simplest model: that water leaks at a rate proportional to its level in the tank. This model is plausible because the pressure at the bottom of the tank is also proportional to the height. The model has the additional merit of generating interesting behavior without generating miserable mathematics.

To formalize this linear-leak model, introduce notation. Let r_1 be the rate at which water leaves the tank. Knowing the dimensions of all the quantities will help in later steps. So it is worth working out the following warmup:

Pause to try 1. What are the dimensions of r_1?

These 'pause to try' questions are designed to help you read actively. Stop and think about the question, work out an answer, and then read onwards to compare your thoughts with the discussion in the text.

The rate r_1 has dimensions of volume per time. Now let's look at other quantities and determine their dimensions. The other quantity that we need is the level in the tank; call it l. The level has dimensions of length. With these symbols, the linear-leak model becomes

$$r_1 \propto l \qquad \text{(linear-leak equation).}$$

Pause to try 2. What are the dimensions of the missing constant of proportionality, which is hidden in the $\propto$ symbol?

The constant of proportionality incorporates the fluid dynamics of the leak (for example, is it viscous or turbulent flow?) and it has dimensions of area per time.

The other part of the model is water conservation. The water that enters either leaves the tank or raises the level. There is no other place for the water to go. Therefore the difference between inflow and outflow changes the water level. Let r_0 be the rate at which water flows into the tank; like r_1, it has dimensions of volume per time. The level in the tank changes because of the net inflow:

$$\dot{l} \propto r_0 - r_1 \qquad \text{(water-conservation equation),}$$

where the notation $\dot{l}$ is a shorthand for dl/dt. The missing constant of proportionality incorporates information about the dimensions of the tank. Here are a few questions for you to check your understanding. Unlike the 'pause to try' questions, the exercises do not have an answer in the following text. They might extend the analysis in the text, or be useful in others way to develop understanding.

Exercise 1. What are the dimensions of the missing constant of proportionality in the water-conservation equation?

Exercise 2. How does increasing the depth of the tank change the constant, if at all?

Exercise 3. How does increasing the width of the tank change the constant, if at all?

Now combine the water-conservation and linear-leak equations. The quickest method is to differentiate the linear-leak equation to get

$$\dot{r}_1 \propto \dot{l}.$$

The water-conservation equation expresses $\dot{l}$ in terms of the inflow and outflow rates, so use that equation to eliminate the level l. The result is

$$\dot{r}_1 \propto r_0 - r_1.$$

Pause to try 3. What are the dimensions of the missing constant of proportionality?

The missing constant has dimensions of inverse time, to match the dt in the denominator of $\dot{r}_1$. Call the constant $1/\tau$. Then τ is called the **time constant** of the system. With the τ the leaky-tank **differential equation** is

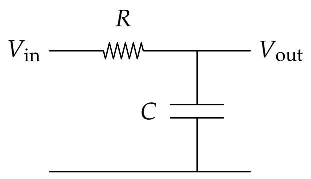
$$\tau\dot{r}_1 = r_0 - r_1$$

or

$$\dot{r}_1 = -\frac{r_1}{\tau} + \frac{r_0}{\tau}.$$

1.1.2 Analogous circuit

You have met this differential equation in previous courses. For example, here is an RC circuit. Its differential equation is



$$\dot{V}_{\text{out}} = -\frac{V_{\text{out}}}{\tau} + \frac{V_{\text{in}}}{\tau},$$

where the time constant τ is RC.

Exercise 4. Show that this differential equation is correct.

The RC equation has the same structure as the leaky-tank equation: The inflow r_0 is like the input voltage V_{in}, and outflow r_1 is like the output voltage V_{out}. Both equations also have a time constant τ, although each constant arises from physics particular to that situation. The time constant in the RC circuit arises from Maxwell's equations of electromagnetism whereas the time constant in the leaky tank arises from the Navier–Stokes equations of fluid mechanics. With that caveat and by making the substitutions between flow rate and voltage, the two systems are identical.

By making this kind of **analogy**, you can use experience with circuits to help you understand mechanical systems and can use your experience with mechanical systems to understand circuits.

1.2 Qualitative understanding

Now that the system is represented as a differential equation, we can play with it to find out how the equation behaves – to obtain a qualitative understanding. We do so by using a simple input and by analyzing the resulting output using a discrete-time approximation simple enough to make with pencil and paper.

1.2.1 Simple input

You can qualitatively understand its behavior by reasoning through putting in a simple input signal r_0, hoping that the output signal r_1 will be easy to understand. Perhaps the simplest input is a **step function**. A step-function input means the input tap has been off forever, turns on at time $t = 0$, and remains at this flow rate r forever. To sketch the output, which is the flow rate $r_1(t)$, use the useful technique of **extreme cases**. Here look at extreme cases of time. Typical extreme cases are $t = -\infty$, 0, and ∞. Nothing of interest happens at $t = -\infty$, so start investigating the next extreme, $t \approx 0$, and find the state of the system. Just at $t = 0$, when the tap turns on, the tank is still empty.

Pause to try 4. Why is the tank empty at $t = 0$?

Since no water has flowed in since $t = -\infty$, all the water in the tank has had time to drain. With zero pressure, the outflow rate $r_1(0)$ is also zero. In this $t \approx 0$ limit, the r_1 term vanishes from the right side of the leaky-tank differential equation, which simplifies to:

$$\dot{r}_1 = \frac{r_0}{\tau} \qquad (t \approx 0).$$

We want to solve for r_1, which is not hard because r_0 is a constant r for $t \geq 0$. The solution is linear growth:

$$r_1(t) = \frac{t}{\tau} r,$$

where r is the constant inflow rate. Although the outflow rate increases linearly in the $t \approx 0$ approximation, it cannot do so forever. Otherwise the outflow would eventually – when $t > \tau$ – exceed the inflow, which would eventually drain the tank. Since an empty tank has no pressure and no outflow yet the outflow is supposed to be greater than r, the situation contradicts itself. So the linear-increase approximation eventually refutes itself. However, like many things in life, the approximation is useful in moderation.

1.2.2 Discrete-time approximation

Rather than waiting for $t > \tau$, we can use the approximation for a short time step Δt. To decide how long Δt should be, think about the tradeoffs. The shorter the Δt, the more accurate is the approximation to the continuous-time system. However, the longer the Δt, the easier are the paper-and-pencil calculations. We compromise between these opposing arguments by using $\Delta t = \tau/2$.

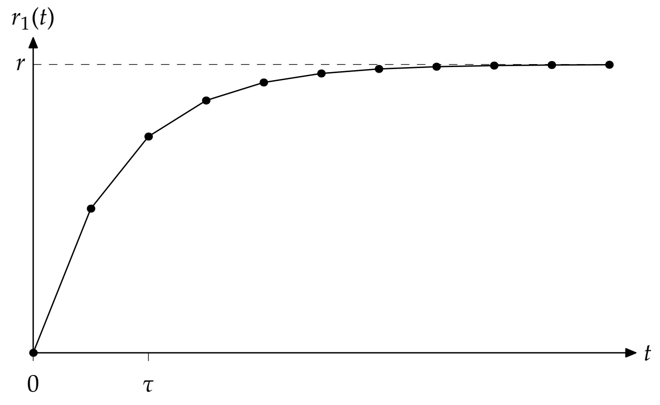
In this approximation, r_1 starts at 0 and builds to $r_1(\tau/2) = r/2$. Then use the leaky-tank differential equation to find the new $\dot{r}_1$ at $t = \tau/2$. The differential equation is

$$\dot{r}_1 = -\frac{r_1}{\tau} + \frac{r_0}{\tau}.$$

For the second time step, which is the range $t = \Delta t \ldots 2\Delta t$, replace r_1 on the right side by its approximate value $r/2$. As usual, r_0 on the right side is replaced by its steady value r. Then

$$\dot{r}_1 = -\frac{1}{2}\frac{r}{\tau} + \frac{r}{\tau} = \frac{1}{2}\frac{r}{\tau}.$$

So r_1 increases at one-half of the steady rate that it increased in the first time step. Let's run the machine for another time step. By the end of that step, r_1 reaches $3r/4$. Following the same procedure, you find that the new $\dot{r}_1$ is $r/4\tau$. Hmmm! The outflow is approaching the constant r and it is increasing ever more slowly. In fact, the gap $r - r_1(t)$ halves with every time step $\Delta t = \tau/2$ but never reaches zero. Here is a sketch of the outflow in this approximation:

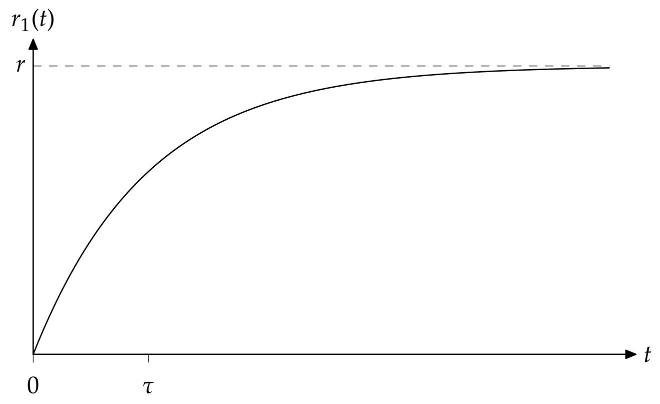


The exact, continuous-time solution for a step-function input is

$$r_1(t) = \begin{cases} 0 & \text{for } t < 0; \\ r(1 - e^{-t/\tau}) & \text{for } t \geq 0. \end{cases}$$

Exercise 5. Check this solution.

Here is a sketch of the exact solution:



Pause to try 5. Compare the discrete-time, approximate solution and the continuous-time solution: Which converges more rapidly to the $t \to \infty$ value $r_1(\infty) = r$?

To compare the approximations, look at their respective values when $t = \tau$. Actually, look at their deviation from the final value $r_1(\infty) = r$. At time τ, the continuous-time solution deviates by r/e, whereas this discrete-time solution deviates by $r/4$. Since $4 > e$, the discrete-time solution overestimates the rate of approach to the steady-state value. However, that inaccuracy is a worthwhile tradeoff to make in a first analysis. With the discrete-time approximation, we can discover important features of the solution using only paper, pencil, and sketches – items that would be available on a desert island.

1.3 Formalizing discrete-time: forward Euler

Let's formalize the discrete-time approximation of the continuous-time differential equation. The steps to determine the next value of r_1 are:

1. Use the already computed $r_1(t)$ and the known input $r_0(t)$ to find $\dot{r}_1(t)$:
$$\dot{r}_1(t) = \frac{r_0(t) - r_1(t)}{\tau}.$$
2. Assume that $\dot{r}_1(t)$ remains constant for the discretization time Δt (in the previous analysis, $\Delta t = \tau/2$).
3. Use that assumption, along with the already computed $r_1(t)$, to find $r_1(t + \Delta t)$:
$$r_1(t + \Delta t) = r_1(t) + \frac{\Delta t}{\tau}(r_0(t) - r_1(t)).$$
4. Go to step 1 with $t + \Delta t$ becoming the new t.

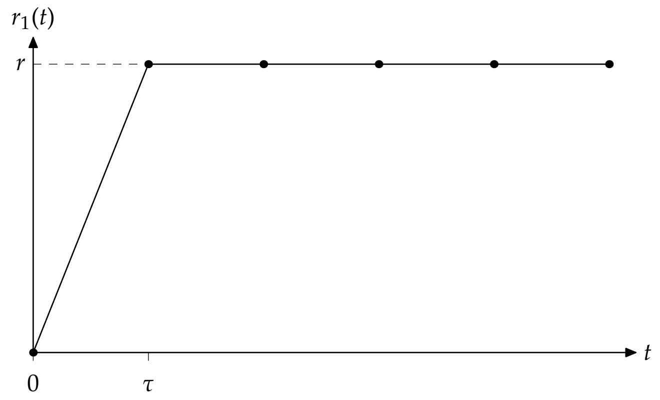
The discrete-time equation is
$$r_1(t + \Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) r_1(t) + \frac{\Delta t}{\tau} r_0(t).$$

This equation is a **difference equation** and is an application of the forward-Euler method. It provides a way for a computer to solve the differential equation approximately. This method also provides a recipe that we can interpret to describe the behavior of the leaky tank. When $\Delta t \approx 0$, the first term on the right is nearly $r_1(t)$: It has weight $1 - \epsilon$ where $\epsilon = \Delta t/\tau$. The second term incorporates $r_0(t)$ with weight ϵ. So the two terms combine into a weighted average of $r_1(t)$ and $r_0(t)$ with weights $1 - \epsilon$ and ϵ, respectively. Therefore the leaky tank, like the RC circuit, outputs a decaying average of its input, which smooths the input. As an example, when the input is the step function, which has a discontinuity, the output is continuous, which is one derivative smoother than the step function.

1.3.1 Extreme cases of the time step

The hand simulation of the step-function response used $\Delta t = \tau/2$. Let's investigate the discrete-time approximation for other values of Δt.

First try $\Delta t = \tau$. The outflow $r_1(t)$ is a straight ramp till $r_1(t) = r$, which happens at the end of the time step when $t = \tau$. Since $r_0(t) = r$ for $t > 0$, the difference between inflow and outflow is zero, so the level does not change, and the outflow is also constant. Thus the approximate solution is:



Now increase Δt to say $\Delta t = 2\tau$. The same initial ramp happens as when $\Delta t = \tau$, but the ramp now ends later at $t = 2\tau$. Then $r_1(2\tau) = 2r$. The leaky-tank differential equation then gives $\dot{r}_1(2\tau) = -r/\tau$. Applying this derivative for one time step $\Delta t = 2\tau$ produces a new flow rate $r_1(2\Delta t) = 0$. From here the cycle repeats. This barely stable oscillation is interesting because the exact solution shows no oscillation. So with $\Delta t = 2\tau$, the discrete-time approximation produces a qualitatively incorrect conclusion.

Now increase Δt to 3τ. The values of r_1 are:

$$\begin{aligned} r_1(0\Delta t) &= 0,\\ r_1(1\Delta t) &= 3r,\\ r_1(2\Delta t) &= -3r,\\ r_1(3\Delta t) &= 9r,\\ r_1(4\Delta t) &= -15r,\\ &\ldots \end{aligned}$$

These oscillations are unstable! This discrete-time approximation gives qualitatively incorrect results, for example suggesting that the water level in the tank, or the rate of outflow, can oscillate unstably despite a steady input. But as we will see later, it takes at least two such systems connected with feedback to create oscillations.

These qualitative inaccuracies suggest trying the other extreme of Δt, that $\Delta t \to 0$. To find how the system behaves, return to the difference equation:

$$r_1(t + \Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) r_1(t) + \frac{\Delta t}{\tau} r_0(t).$$

It is easiest to solve if we make a few changes. First, define

$$y[n] \equiv r_1(n\Delta t).$$

The time-step number n is a dimensionless measure of time. With $r_0(t) = r$, the leaky-tank difference equation is

$$y[n+1] = (1-\epsilon) y[n] + \epsilon r,$$

where $\epsilon = \Delta t/\tau$. Now make a dimensionless measure of rate: $Q = Y/r$. This division is an example of a useful technique: **taking out the big part**. Then

$$q[n+1] = (1-\epsilon) q[n] + \epsilon.$$

Now take out the big part again by measuring the rate relative to its long-term, steady-state value of r, which here is $q[\infty] = 1$. So define $P = 1 - Q$. Then

$$1 - p[n+1] = (1-\epsilon)(1-p[n]) + \epsilon,$$

or

$$p[n+1] = (1-\epsilon)p[n].$$

With the boundary condition $p[0] = 1$, the solution is

$$p[n] = (1-\epsilon)^n.$$

Invert all the changes of variable to solve for $r_1(t)$:

$$r_1(n\Delta t) = r\left(1 - (1-\epsilon)^n\right).$$

Now use the approximation that $\Delta t \to 0$, so $\epsilon = \Delta t/\tau \to 0$ as well. Then

$$(1-\epsilon)^n \approx e^{-n\epsilon}.$$

Exercise 6. Justify this approximation for e^x.

Exercise 7. In a genetics experiment, 95% of bacteria are mutated with every treatment. After 140 treatments, what fraction were never mutated? It is worth practicing such questions until you can answer to within a factor of 3 in less than 30 seconds. [The data are from an experiment described in a biology seminar at Caltech in 1995.]

In this $\epsilon \to 0$ limit,

$$r_1(n\Delta t) = (1 - e^{-n\Delta t/\tau})r.$$

Call $t = n\Delta t$ and get

$$r_1(t) = (1 - e^{-t/\tau})r,$$

which is the exact solution from integrating the continuous-time, differential equation.

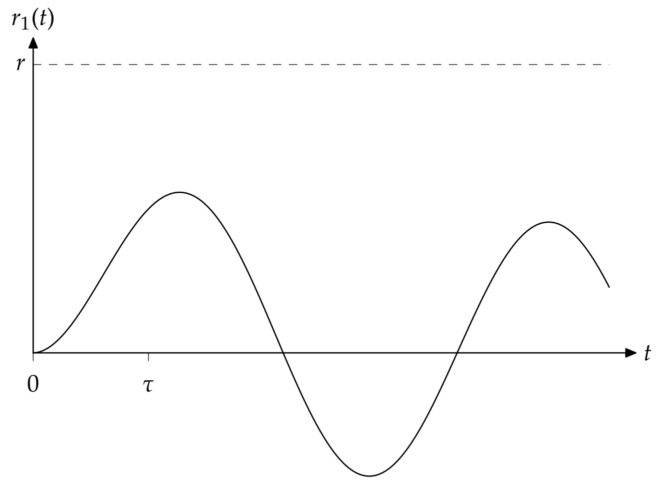
1.4 Heat storage in the ocean

An ocean is a leaky-tank system! Heat flows into the ocean from the sun, land, and air. Heat inflow is proportional to the land or air temperature (leaving out the sun for simplicity). Heat leaks from the ocean into the surroundings at a rate proportional to the ocean temperature. So the ocean temperature is like the tank level; the input signal is the land or air temperature, which we approximate as the same; and the output signal is the ocean temperature. With this analogy, we can simulate the water temperature by using the leaky-tank equations.

The only parameter of a leaky-tank system is its time constant. The ocean's time constant is perhaps $\tau = 3$ months, because it can store a lot of heat. The input temperature $T_0(t)$, produced by the ground and air warmed by the sun, oscillates with a period of 6 months.

Pause to try 6. Simulate the system to find the output temperature $T_1(t)$ and discuss any interesting features of the result.

To simulate the output temperature $T_1(t)$, use the forward-Euler method. Here is a simulation for when the input signal is zero for $t < 0$ and starts oscillating at $t = 0$:



The output oscillates, which is surprising. When you look at the equation, it is not obvious that a sinusoidal input will give a sinusoidal output. But the oscillating output makes sense if you have swum at the beach during several seasons: winter is colder than spring, which is colder than summer, etc.

We can also find the temperature by solving the differential equation in closed form. When the system is given an eternal sinusoidal input $T_0(t) = A\sin\beta t$ (including for $t < 0$), the closed-form solution is

$$T_1(t) = \frac{A}{\sqrt{1 + (\beta\tau)^2}}\sin(\beta t - \phi),$$

where $\phi = \arctan\beta\tau$ is the so-called **phase lag**.

Exercise 8. Check this solution.

This form shows two features of the output temperature:

- **diminution**: Because of the $\sqrt{1 + (\beta\tau)^2}$ factor in the denominator, which is greater than 1, the output signal is shrunken compared to the input signal. Therefore ocean temperature fluctuates less than land temperature, and the ocean is more hospitable for life to evolve.
- **delay**: Because of the $-\phi$ contribution to the phase, the output is delayed relative to the input. So, although the sun's peak temperature input is in June at the summer solstice, the ocean

temperature peaks a few months later. The best time to swim or surf in the Massachusetts waters is in August!

These two behaviors, which are canonical for a first-order system such as a tank, will be revisited when we study signal processing and we will build intuitions for each behavior.

Exercise 9. Why does an unheated house feel coldest at around, say, 4am rather than at midnight, even though more sunlight is available at 4am to heat the ground and house?

Recitation 2
Difference equations and modularity

The goals of this chapter are:

- to illustrate modularity and to describe systems in a modular way;
- to translate problems from their representation as a verbal description into their representation as discrete-time mathematics (difference equations); and
- to start investigating the simplest second-order system, the second-simplest module for analyzing and designing systems.

The themes of this chapter are modularity and the **representation** of verbal descriptions as discrete-time mathematics. We illustrate these themes with two examples, money in a hypothetical MIT endowment fund and rabbits reproducing in a pen, setting up difference equations to represent them. The rabbit example, which introduces a new module for building and analyzing systems, is a frequent visitor to these chapters. In this chapter we begin to study how that module behaves. Before introducing the examples, we illustrate what modularity is and why it is useful.

2.1 Modularity: Making the input like the output

A common but alas non-modular way to formulate difference and differential equations uses boundary conditions. An example from population growth illustrates this formulation and how to improve it by making it modular. The example is the population of the United States. The US population grows at an annual rate of roughly 1%, according to the *World FactBook* [1], and the US population is roughly 300 million in 2007. What will be the US population be in 2077 if the growth rate remains constant at 1%?

Pause to try 7. What is the population equation and boundary condition representing this information?

The difference equation for the population in year n is

$$p[n] = (1 + r)p[n-1] \qquad \text{(population equation)},$$

where $r = 0.01$ is the annual growth rate. The boundary condition is

$$p[2007] = 3 \times 10^8 \qquad \text{(boundary condition)}.$$

To find the population in 2077, solve this difference equation with boundary condition to find $p[2077]$.

Exercise 10. What is p[2077]? How could you have quickly approximated the answer?

You might wonder why, since no terms are subtracted, the population equation is called a difference equation. The reason is by analogy with differential equations, which tell you how to find $f(t)$ from $f(t - \Delta t)$, with Δt going to 0. Since the discrete-time population equation tells us how to find $f[n]$ from $f[n-1]$, it is called a difference equation and its solution is the subject of the calculus of finite differences. When the goal – here, the population – appears on the input side, the difference equation is also a **recurrence relation**. What recurrence has to do with it is the topic of an upcoming chapter; for now take it as pervasive jargon.

The mathematical formulation as a recurrence relation with boundary condition, while sufficient for finding $p[2077]$, is messy: The boundary condition is a different kind of object from the solution to a recurrence. This objection to clashing categories may seem philosophical – in the colloquial meaning of philosophical as irrelevant – but answering it helps us to understand and design systems. Here the system is the United States. The input to the system is one number, the initial population $p[2007]$; however, the output is a sequence of populations $p[2008], p[2009], \ldots$. In this formulation, the system's output cannot become the input to another system. Therefore we cannot design large systems by combining small, easy-to-understand systems. Nor we can we analyze large, hard-to-understand systems by breaking them into small systems.

Instead, we would like a modular formulation in which the input is the same kind of object as the output. Here is the US-population question reformulated along those lines: *If $x[n]$ people immigrate into the United states in year n, and the US population grows at 1% annually, what is the population in year n?* The input signal is the number of immigrants versus time, so it is a sequence like the output signal. Including the effect of immigration, the recurrence is

$$\underbrace{p[n]}_{\text{output}} = \underbrace{(1+r)p[n-1]}_{\text{reproduction}} + \underbrace{x[n]}_{\text{immigration}} .$$

The boundary condition is no longer separate from the equation! Instead it is part of the input signal. This modular formulation is not only elegant; it is also more general than is the formulation with boundary conditions, for we can recast the original question into this framework. The recasting involves finding an input signal – here the immigration versus time – that reproduces the effect of the boundary condition $p[2007] = 3 \times 10^8$.

Pause to try 8. What input signal reproduces the effect of the boundary condition?

The boundary condition can be reproduced with this immigration schedule (the input signal):

$$x[n] = \begin{cases} 3 \times 10^8 & \text{if } n = 2007; \\ 0 & \text{otherwise.} \end{cases}$$

This model imagines an empty United States into which 300 million people arrive in the year 2007. The people grow (in numbers!) at an annual rate of 1%, and we want to know $p[2077]$, the output signal (the population) in the year 2077.

The general formulation with an arbitrary input signal is harder to solve directly than is the familiar formulation using boundary conditions, which can be solved by tricks and guesses. For our input signal, the output signal is

$$p[n] = \begin{cases} 3 \cdot 10^8 \times 1.01^{n-2007} & \text{for } n \geq 2007; \\ 0 & \text{otherwise.} \end{cases}$$

Exercise 11. Check that this output signal satisfies the boundary condition and the population equation.

In later chapters you learn how to solve the formulation with an arbitrary input signal. Here we emphasize not the method of solution but the modular formulation where a system turns one signal into another signal. This modular description using signals and systems helps analyze complex problems and build complex systems.

To see how it helps, first imagine a world with two countries: Ireland and the United States. Suppose that people emigrate from Ireland to the United States, a reasonable model in the 1850's. Suppose also that the Irish population has an intrinsic 10% annual decline due to famines and that another 10% of the population emigrate annually to the United States. Ireland and the United States are two systems, with one system's output (Irish emigration) feeding into the other system's input (the United States's immigration). The modular description helps when programming simulations. Indeed, giant population-growth simulations are programmed in this object-oriented way. Each system is an object that knows how it behaves – what it outputs – when fed input signals. The user selects systems and specifies connections among them. Fluid-dynamics simulations use a similar approach by dividing the fluid into zillions of volume elements. Each element is a system, and energy, entropy, and momentum emigrate between neighboring elements.

Our one- or two-component population systems are simpler than fluid-dynamics simulations, the better to illustrate modularity. Using two examples, we next practice modular description and how to represent verbal descriptions as mathematics.

2.2 Endowment gift

The first example for representing descriptions as mathematics involves a hypothetical endowment gift to MIT. A donor gives $\$10^7$ dollars to MIT to support projects proposed and chosen by MIT undergraduates! MIT would like to use this fund for a long time and draw $\$0.5 \times 10^6$ every year for a so-called 5% drawdown. Assume that the money is placed in a reliable account earning 4% interest compounded annually. How long can MIT and its undergraduates draw on the fund before it dwindles to zero?

Never make a calculation until you know roughly what the answer will be! This maxim is recommended by John Wheeler, a brilliant physicist whose most famous student was MIT alum Richard Feynman [2]. We highly recommend Wheeler's maxim as a way to build intuition. So here are a few estimation questions to get the mental juices flowing. Start with the broadest distinction, whether a number is finite or infinite. This distinction suggests the following question:

Pause to try 9. Will the fund last forever?

Alas, the fund will not last forever. In the first year, the drawdown is slightly greater than the interest, so the endowment capital will dwindle slightly. As a result, the next year's interest will be smaller than the first year's interest. Since the drawdown stays the same at $500,000 annually (which is 5% of the initial amount), the capital will dwindle still more in later years, reducing the interest, leading to a greater reduction in interest, leading to a greater reduction in capital... Eventually the fund evaporates. Given that the lifetime is finite, roughly how long is it? Can your great-grandchildren use it?

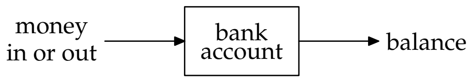
Pause to try 10. Will the fund last longer than or shorter than 100 years?

The figure of 100 years comes from the difference between the outflow – the annual drawdown of 5% of the gift – and the inflow produced by the interest rate of 4%. The difference between 5% and 4% annually is $\delta = 0.01/\text{year}$. The dimensions of δ are inverse time, suggesting an endowment lifetime of $1/\delta$, which is 100 years. Indeed, if every year were like the first, the fund would last for 100 years. However, the inflow from interest decreases as the capital decreases, so the gap between outflow and inflow increases. Thus this $1/\delta$ method, based on extrapolating the first year's change to every year, overestimates the lifetime.

Having warmed up with two estimates, let's describe the system mathematically and solve for the true lifetime. In doing so, we have to decide what is the input signal, what is the output signal, and what is the system. The system is the least tricky part: It is the bank account paying 4% interest. The gift of $10 million is most likely part of the input signal.

Pause to try 11. Is the $500,000 annual drawdown part of the output or the input signal?

The drawdown flows out of the account, and the account is the system, so perhaps the drawdown is part of the output signal. No!! The output signal is what the system does, which is to produce or at least to compute a balance. The input signal is what you do to the system. Here, you move money in or out of the system:



The initial endowment is a one-time positive input signal, and the annual drawdown is a recurring negative input signal. To find how long the endowment lasts, find when the output signal crosses below zero. These issues of representation are helpful to figure out *before* setting up mathematics. Otherwise with great effort you create irrelevant equations, whereupon no amount of computing power can help you.

Now let's represent the description mathematically. First represent the input signal. To minimize the large numbers and dollar signs, measure money in units of $500,000. This choice makes the input signal dimensionless:

$$X = 20, -1, -1, -1, -1, \ldots$$

We use the notation that a capital letter represents the entire signal, while a lowercase letter with an index represents one sample from the signal. For example, P is the sequence of populations and $p[n]$ is the population in year n.

The output signal is

$$Y = 20, ?, ?, ?, \ldots$$

Pause to try 12. Explain why $y[0] = 20$.

The problem is to fill in the question marks in the output signal and find when it falls below zero. The difference equation describing the system is

$$y[n] = (1 + r)y[n - 1] + x[n],$$

where r is the annual interest rate (here, $r = 0.04$). This difference equation is a first-order equation because any output sample $y[n]$ depends on the one preceding sample $y[n - 1]$. The system that the equation represents is said to be a first-order system. It is the simplest module for building and analyzing complex systems.

Exercise 12. Compare this equation to the one for estimating the US population in 2077.

Now we have formulated the endowment problem as a signal processed by a system to produce another signal – all hail modularity! – and represented this description mathematically. However, we do not yet know how to solve the mathematics for an arbitrary input signal X. But here we need to solve it only for the particular input signal

$$X = 20, -1, -1, -1, -1, \ldots.$$

With that input signal, the recurrence becomes

$$y[n] = \begin{cases} 1.04 \cdot y[n-1] - 1 & n > 0; \\ 20 & n = 0. \end{cases}$$

The $y[0] = 20$ reflects that the donor seeds the account with 20 units of money, which is the \$10,000,000 endowment. The -1 in the recurrence reflects that we draw 1 unit every year. Without the -1 term, the solution to the recurrence would be $y[n] \sim 1.04^n$, where the $\sim$ symbol means 'except for a constant'. The -1 means that simple exponential growth is not a solution. However, -1 is a constant so it may contribute only a constant to the solution. That reasoning is dubious but simple, so try it first. Using a bit of courage, here is a guess for the form of the solution:

$$y[n] = A \cdot 1.04^n + B \qquad \text{(guess)},$$

where A and B are constants to be determined. Before finding A and B, figure out the most important characteristic, their signs. So:

Pause to try 13. Assume that this form is correct. What are the signs of A and B?

Since the endowment eventually vanishes, the variable term $A \cdot 1.04^n$ must make a negative contribution; so $A < 0$. Since the initial output $y[0]$ is positive, B must overcome the negative contribution from A; so $B > 0$.

Pause to try 14. Find A and B.

Solving for two unknowns A and B requires two equations. Each equation will probably come from one condition. So match the guess to the known balances at two times. The times (values of n) that involve the least calculation are the extreme cases $n = 0$ and $n = 1$. Matching the guess to the behavior at $n = 0$ gives the first equation:

$$20 = A + B \qquad (n = 0 \text{ condition}).$$

To match the guess to the behavior at $n = 1$, first find $y[1]$. At $n = 1$, which is one year after the gift, 0.8 units of interest arrive from 4% of 20, and 1 unit leaves as the first drawdown. So

$$y[1] = 20 + 0.8 - 1 = 19.8.$$

Matching this value to the guess gives the second equation:

$$19.8 = 1.04A + B \qquad (n = 1 \text{ condition}).$$

Both conditions are satisfied when $A = -5$ and $B = 25$. As predicted, $A < 0$ and $B > 0$. With that solution the guess becomes

$$y[n] = 25 - 5 \times 1.04^n.$$

This solution has a strange behavior. After the balance drops below zero, the 1.04^n grows ever more rapidly so the balance becomes negative ever faster.

Exercise 13. Does that behavior of becoming negative more and more rapidly indicate an incorrect solution to the recurrence relation, or an incomplete mathematical translation of what happens in reality?

Exercise 14. The guess, with the given values for A and B, works for $n = 0$ and $n = 1$. (How do you know?) Show that it is also correct for $n > 1$.

Now we can answer the original question: When does $y[n]$ fall to zero? Answer: When $1.04^n > 5$, which happens at $n = 41.035\ldots$. So MIT can draw on the fund in years $1, 2, 3, \ldots, 41$, leaving loose change in the account for a large graduation party. The exact calculation is consistent with the argument that the lifetime be less than 100 years.

Exercise 15. How much loose change remains after MIT draws its last payment? Convert to real money!

2.3 Rabbits

The second system to represent mathematically is the fecundity of rabbits. The *Encyclopedia Britannica* (1981 edition) states this population-growth problem as follows [3]:

> A certain man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begs a new pair which from the second month on becomes productive?

That description is an English representation of the original Latin. We first represent the verbal description mathematically and then play with the equations to understand how the system behaves. It is the simplest system beyond the first-order systems like the endowment, so it is an important module for building and analyzing complex systems.

2.3.1 From words to recurrence

Before representing the system mathematically, we describe it modularly using signals and systems by finding a system, an input signal, and an output signal. It is usually easiest to begin by looking for the system since it is the active element. The phrase 'surrounding on all sides by a wall' indicates a candidate for a system. The system is the inside of the wall, which is where the rabbits reproduce, together with the rules under which rabbits reproduce.

Pause to try 15. What is the input signal?

An input to the system is placing rabbits into it or taking them from it. The input signal is the number of pairs that enter the system at month n, where the signal would be negative if rabbits emigrate from the system to seek out tastier grass or other rabbit friends.

Pause to try 16. What is the output signal?

Some pairs are placed into the system as children (the immigrants); other pairs are born in the system (the native born). The sum of these kinds of pairs is the output signal.

To describe the system mathematically, decompose it by type of rabbit:

1. **children**, who cannot reproduce but become adults in one month; and
2. **adults**, who reproduce that month and thereafter.

Let $c[n]$ be the number of child pairs at month n and $a[n]$ be the number of adult pairs at month n. These intermediate signals combine to make the output signal:

$$f[n] = a[n] + c[n] \qquad \text{(output signal).}$$

Pause to try 17. What equation contains the rule that children become adults in one month?

Because children become adults in one month, and adults do not die, the pool of adults grows by the number of child pairs in the previous month:

$$a[n] = a[n-1] + c[n-1] \qquad \text{(growing-up equation)}.$$

The two terms on the right-hand side represent the two ways to be an adult:

1. You were an adult last month ($a[n-1]$), or
2. you were a child last month ($c[n-1]$) and grew up.

The next equation says that all adults, and only adults, reproduce to make new children:

$$c[n] = a[n-1].$$

However, this equation is not complete because immigration also contributes child pairs. The number of immigrant pairs at month n is the input signal $x[n]$. So the full story is:

$$c[n] = a[n-1] + x[n] \qquad \text{(child equation)}$$

Our goal is a recurrence for $f[n]$, the total number of pairs. So we eliminate the number of adult pairs $a[n]$ and the number of child pairs $c[n]$ in favor of $f[n]$. Do it in two steps. First, use the growing-up equation to replace $a[n-1]$ in the child equation with $a[n-2]+c[n-2]$. That substitution gives

$$c[n] = a[n-2] + c[n-2] + x[n].$$

Since $f[n] = c[n] + a[n]$, we can turn the left side into $f[n]$ by adding $a[n]$. The growing-up equation says that $a[n]$ is also $a[n-1] + c[n-1]$, so add those terms to the right side and pray for simplification. The result is

$$\underbrace{c[n] + a[n]}_{f[n]} = \underbrace{a[n-2] + c[n-2]}_{f[n-2]} + x[n] + \underbrace{a[n-1] + c[n-1]}_{f[n-1]}.$$

The left side is $f[n]$. The right side contains $a[n-2]+c[n-2]$, which is $f[n-2]$; and $a[n-1]+c[n-1]$, which is $f[n-1]$. So the sum of equations simplifies to

$$f[n] = f[n-1] + f[n-2] + x[n].$$

The Latin problem description is from Fibonacci's *Liber Abaci* [4], published in 1202, and this equation is the famous Fibonacci recurrence but with an input signal $x[n]$ instead of boundary conditions.

This mathematical representation clarifies one point that is not obvious in the verbal representation: The number of pairs of rabbits at month n depends on the number in months $n-1$ and $n-2$. Because of this dependence on two preceding samples, this difference equation is a second-order difference equation. Since all the coefficients are unity, it is the simplest equation of that category, and ideal as a second-order system to understand thoroughly. To build that understanding, we play with the system and see how it responds.

2.3.2 Trying the recurrence

To play with the system described by Fibonacci, we need to represent Fibonacci's boundary condition that one pair of child rabbits enter the walls only in month 0. The corresponding input signal is $X = 1, 0, 0, 0, \ldots$. Using that X, known as an **impulse** or a **unit sample**, the recurrence produces (leaving out terms that are zero):

$$\begin{aligned} f[0] &= x[0] = 1, \\ f[1] &= f[0] = 1, \\ f[2] &= f[0] + f[1] = 2, \\ f[3] &= f[1] + f[2] = 3, \\ &\ldots \end{aligned}$$

When you try a few more lines, you get the sequence: $F = 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$. When you tire of hand calculation, ask a computer to continue. Here is slow Python code to print $f[0]$, $f[1]$,...,$f[19]$:

```
def f(n):
  if n < 2: return 1
  return f(n-1) + f(n-2)
print [f(i) for i in range(20)]
```

Exercise 16. Write the corresponding Matlab or Octave code, then rewrite the code in one of the languages – Python, Matlab, or Octave – to be efficient.

Exercise 17. Write Matlab, Octave, or Python code to find $f[n]$ when the input signal is $1, 1, 1, \ldots$. What is $f[17]$?

2.3.3 Rate of growth

To solve the recurrence in **closed form** – meaning an explicit formula for $f[n]$ that does not depend on preceding samples – it is helpful to investigate its approximate growth. Even without sophisticated techniques to find the output signal, we can understand the growth in this case when the input signal is the impulse.

Pause to try 18. When the input signal is the impulse, how fast does $f[n]$ grow? Is it polynomial, logarithmic, or exponential?

From looking at the first few dozen values, it looks like the sequence grows quickly. The growth is almost certainly too rapid to be logarithmic and, almost as certain, too fast to be polynomial unless it is a high-degree polynomial. Exponential growth is the most likely candidate, meaning that an approximation for $f[n]$ is

$$f[n] \sim z^n$$

where z is a constant. To estimate z, play with the recurrence when $n > 0$, which is when the input signal is zero. The $f[n]$ are all positive and, since $f[n] = f[n-1] + f[n-2]$ when $n > 0$, the samples are increasing: $f[n] > f[n-1]$. This bound turns $f[n] = f[n-1] + f[n-2]$ into the inequality

$$f[n] < f[n-1] + f[n-1].$$

So $f[n] < 2f[n-1]$ or $f[n]/f[n-1] < 2$; therefore the upper bound on z is $z < 2$. This bound has a counterpart lower bound obtained by replacing $f[n-1]$ by $f[n-2]$ in the Fibonacci recurrence. That substitution turns $f[n] = f[n-1] + f[n-2]$ into

$$f[n] > f[n-2] + f[n-2].$$

The right side is $2f[n-2]$ so $f[n] > 2f[n-2]$. This bound leads to a lower bound: $z^2 > 2$ or $z > \sqrt{2}$. The range of possible z is then

$$\sqrt{2} < z < 2.$$

Let's check the bounds by experiment. Here is the sequence of ratios $f[n]/f[n-1]$ for $n = 1, 2, 3, \ldots$:

$$1.0, 2.0, 1.5, 1.666\ldots, 1.6, 1.625, 1.615\ldots, 1.619\ldots, 1.617\ldots$$

The ratios seem to oscillate around 1.618, which lies between the predicted bounds $\sqrt{2}$ and 2. In later chapters, using new mathematical representations, you learn how to find the closed from for $f[n]$. We have walked two steps in that direction by representing the system mathematically and by investigating how $f[n]$ grows.

Exercise 18. Use a more refined argument to improve the upper bound to $z < \sqrt{3}$.

Exercise 19. Does the number 1.618 look familiar?

Exercise 20. [Hard!] Consider the same system but with one rabbit pair emigrating into the system every month, not only in month 0. Compare the growth with Fibonacci's problem, where one pair emigrated in month 0 only. Is it now faster than exponential? If yes, how fast is it? If no, does the order of growth change from $z \approx 1.618$?

Recitation 3

Block diagrams and operators: Two new representations

The goals of this chapter are:

- to introduce two representations for discrete-time systems: block diagrams and operators;
- to introduce the whole-signal abstraction and to exhort you to use abstraction;
- to start manipulating operator expressions;
- to compare operator with difference-equation and block-diagram manipulations.

The preceding chapters explained the verbal-description and difference-equation representations. This chapter continues the theme of multiple representations by introducing two new representations: block diagrams and operators. New representations are valuable because they suggest new thoughts and often provide new insight; an expert engineer values her representations the way an expert carpenter values her tools. This chapter first introduces block diagrams, discusses the whole-signal abstraction and the general value of abstraction, then introduces the operator representation.

3.1 Disadvantages of difference equations

Chapter 2 illustrated the virtues of difference equations. When compared to the verbal description from which they originate, difference equations are compact, easy to analyze, and suited to computer implementation. Yet analyzing difference equations often involves chains of micro-manipulations from which insight is hard to find. As an example, show that the difference equation

$$d[n] = a[n] - 3a[n-1] + 3a[n-2] - a[n-3]$$

is equivalent to this set of equations:

$$\begin{aligned} d[n] &= c[n] - c[n-1] \\ c[n] &= b[n] - b[n-1] \\ b[n] &= a[n] - a[n-1]. \end{aligned}$$

As the first step, use the last equation to eliminate $b[n]$ and $b[n-1]$ from the $c[n]$ equation:

$$c[n] = \underbrace{(a[n] - a[n-1])}_{b[n]} - \underbrace{(a[n-1] - a[n-2])}_{b[n-1]} = a[n] - 2a[n-1] + a[n-2].$$

Use that result to eliminate $c[n]$ and $c[n-1]$ from the $d[n]$ equation:

$$\begin{aligned} d[n] &= \underbrace{(a[n] - 2a[n-1] + a[n-2])}_{c[n]} - \underbrace{(a[n-1] - 2a[n-2] + a[n-3])}_{c[n-1]} \\ &= a[n] - 3a[n-1] + 3a[n-2] - a[n-3]. \end{aligned}$$

Voilà: The three-equation system is equivalent to the single difference equation. But what a mess. Each step is plausible yet the chain of steps seems random. If the last step had produced

$$d[n] = a[n] - 2a[n-1] + 2a[n-2] - a[n-3],$$

it would not immediately look wrong. We would like a representation where it would look wrong, perhaps not immediately but at least quickly. Block diagrams are one such representation.

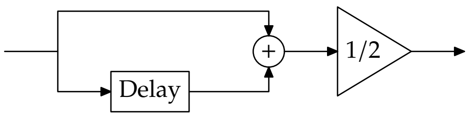
Exercise 21. Although this section pointed out a disadvantage of difference equations, it is also important to appreciate their virtues. Therefore, invent a verbal description (a story) to represent the single equation

$$d[n] = a[n] - 3a[n-1] + 3a[n-2] - a[n-3]$$

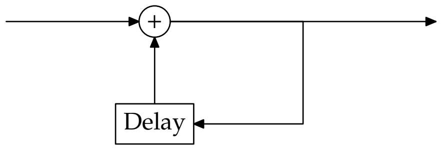
and then a verbal description to represent the equivalent set of three equations. Now have fun showing, without converting to difference equations, that these descriptions are equivalent!

3.2 Block diagrams to the rescue

Block diagrams visually represent a system. To show how they work, here are a few difference equations with corresponding block diagrams:



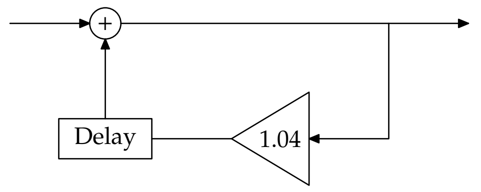
$y[n] = (x[n] + x[n-1])/2$
averaging filter



$y[n] = y[n-1] + x[n]$
account with 0% interest

Pause to try 19. Draw the block diagram for the endowment account from **Section 2.2**.

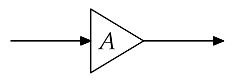
The endowment account is a bank account that pays 4% interest, so it needs a gain element in the loop, with gain equal to 1.04. The diagram is not unique. You can place the gain element before or after the delay. Here is one choice:



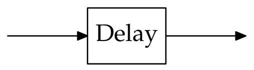
$$y[n] = 1.04\,y[n-1] + x[n]$$

endowment account from **Section 2.2**

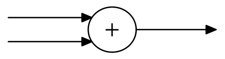
Amazingly, all systems in this course can be built from only two actions and one combinator:



action 1: multiply by α



action 2: delay one tick



combinator: add inputs

3.2.1 Block diagram for the Fibonacci system

To practice block diagrams, we translate (represent) the Fibonacci system into a block diagram.

Pause to try 20. Represent the Fibonacci system of **Section 2.3** using a block diagram.

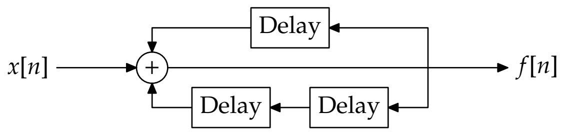
We could translate Fibonacci's description (**Section 2.3**) directly into a block diagram, but we worked so hard translating the description into a difference equation that we start there. Its difference equation is

$$f[n] = f[n-1] + f[n-2] + x[n],$$

where the input signal $x[n]$ is how many pairs of child rabbits enter the system at month n, and the output signal $f[n]$ is how many pairs of rabbits are in the system at month n. In the block diagram, it is convenient to let input signals flow in from the left and to let output signals exit at the right – following the left-to-right reading common to many languages.

Exercise 22. Do signals-and-systems textbooks in Hebrew or Arabic, which are written right to left, put input signals on the right and output signals on the left?

The Fibonacci system combines the input sample, the previous output sample, and the second-previous output sample. These three signals are therefore inputs to the plus element. The previous output sample is produced using a delay element to store samples for one time tick (one month) before sending them onward. The second-previous output sample is produced by using two delay elements in series. So the block diagram of the Fibonacci system is



3.2.2 Showing equivalence using block diagrams

We introduced block diagrams in the hope of finding insight not easily visible from difference equations. So use block diagrams to redo the proof that

$$d[n] = a[n] - 3a[n-1] + 3a[n-2] - a[n-3]$$

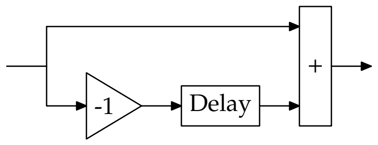
is equivalent to

$$\begin{aligned} d[n] &= c[n] - c[n-1], \\ c[n] &= b[n] - b[n-1], \\ b[n] &= a[n] - a[n-1]. \end{aligned}$$

The system of equations is a cascade of three equations with the structure

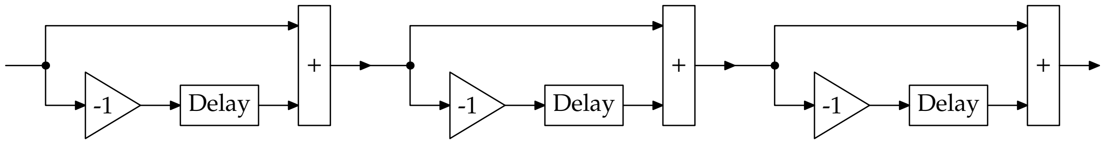
output = this input − previous input.

The block diagram for that structure is



where the gain of −1 produces the subtraction.

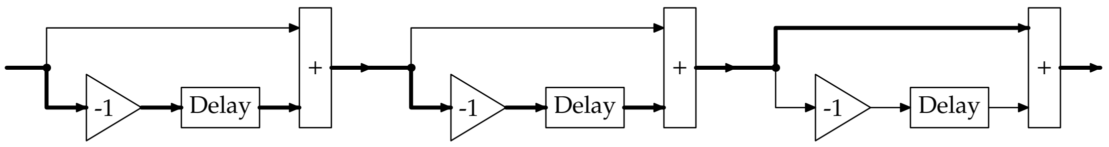
The cascade of three such structures has the block diagram



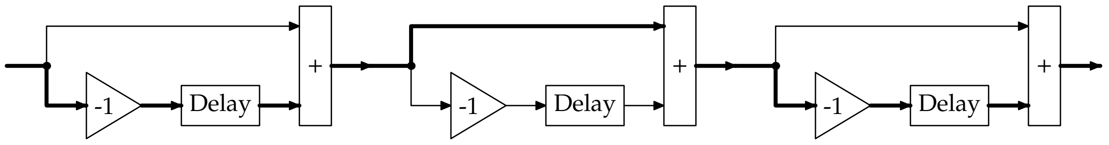
This diagram has advantages compared to the set of difference equations. First, the diagram helps us describe the system compactly. Each stage in the cascade is structurally identical, and the structural identity is apparent by looking at it. Whereas in the difference-equation representation, the common structure of the three equations is hidden by the varying signal names. Each stage, it turns out, is a discrete-time differentiator, the simplest discrete-time analog of a continuous-time differentiator. So the block diagram makes apparent that the cascade is a discrete-time triple differentiator.

Second, the block diagram helps rewrite the system, which we need to do to show that it is identical to the single difference equation. So follow a signal through the cascade. The signal reaches a fork three times (marked with a dot), and each fork offers a choice of the bottom or top branch. Three two-way branches means 2^3 or 8 paths through the system. Let's examine a few of them. Three paths accumulate two delays:

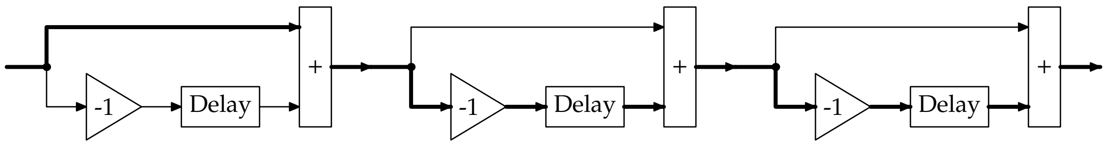
1. low road, low road, high road:



2. low road, high road, low road:



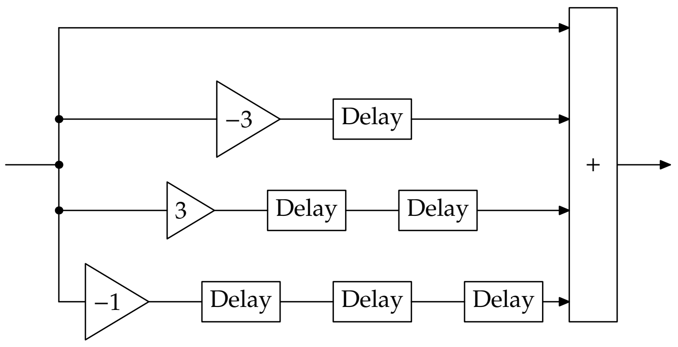
3. high road, low road, low road:



Besides the two delays, each path accumulates two gains of −1, making a gain of 1. So the sum of the three paths is a gain of 3 and a double delay.

Exercise 23. Show the other five paths are: three paths with a single delay and a gain of −1, one path with three delays and a gain of −1, and one path that goes straight through (no gain, no delay).

A block diagram representing those four groups of paths is



The single difference equation

$$d[n] = a[n] - 3a[n-1] + 3a[n-2] - a[n-3].$$

also has this block diagram.

The pictorial approach is an advantage of block diagrams because humans are sensory beings and vision is an important sense. Brains, over hundreds of millions of years of evolution, have developed extensive hardware to process sensory information. However, analytical reasoning and symbol manipulation originate with language, skill perhaps 100,000 years old, so our brains have much less powerful hardware in those domains. Not surprisingly, computers are far more skilled than are humans at analytical tasks like symbolic algebra and integration, and humans are far more skilled than are computers at perceptual tasks like recognizing faces or speech. When you solve problems, amplify your intelligence with a visual representation such as block diagrams.

On the other side, except by tracing and counting paths, we do not know to manipulate block diagrams; whereas analytic representations lend themselves to transformation, an important property when redesigning systems. So we need a grammar for block diagrams. To find the rules of this grammar, we introduce a new representation for systems, the operator representation. This representation requires the whole-signal abstraction in which all samples of a signal combine into one signal. It is a subtle change of perspective, so we first discuss the value of abstraction in general, then return to the abstraction.

3.3 The power of abstraction

Abstraction is a great tools of human thought. All language is built on it: When you use a word, you invoke an abstraction. The word, even an ordinary noun, stands for a rich, subtle, complex idea. Take *cow* and try to program a computer to distinguish cows from non-cows; then you find how difficult abstraction is. Or watch a child's ability with language develop until she learns that 'red' is not a property of a particular object but is an abstract property of objects. No one knows how the mind manages these amazing feats, nor – in what amounts to the same ignorance – can anyone teach them to a computer.

Abstraction is so subtle that even Einstein once missed its value. Einstein formulated the theory of special relativity [5] with space and time as separate concepts that mingle in the Lorentz transformation. Two years later, the mathematician Hermann Minkowski joined the two ideas into the spacetime abstraction:

> The views of space and time which I wish to lay before you have sprung from the soil of experimental physics, and therein lies their strength. They are radical. Henceforth space by itself, and time by itself, are doomed to fade away into mere shadows, and only a kind of union of the two will preserve an independent reality.

See the English translation in [6] or the wonderful textbook *Spacetime Physics* [7], whose first author recently retired from the MIT physics department. Einstein thought that spacetime was a preposterous invention of mathematicians with time to kill. Einstein made a mistake. It is perhaps the fundamental abstraction of modern physics. The moral is that abstraction is powerful but subtle.

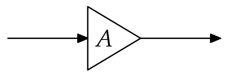
Exercise 24. Find a few abstractions in chemistry, biology, physics, and programming.

If we lack Einstein's physical insight, we ought not to compound the absence with his mistake. So look for and create abstractions. For example, in a program, factor out common code into a procedure and encapsulate common operations into a class. In general, organize knowledge into abstractions or chunks [8].

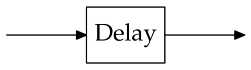
3.4 Operations on whole signals

For signals and systems, the whole-signal abstraction increases our ability to analyze and build systems. The abstraction is take all samples of a signal and lump them together, operating on the entire signal at once and as one object. We have not been thinking that way because most of our representations hinder this view. Verbal descriptions and difference equations usually imply a sample-by-sample analysis. For example, for the Fibonacci recurrence in Section 2.3.2, we found the zeroth sample $f[0]$, used $f[0]$ to find $f[1]$, used $f[0]$ and $f[1]$ to find $f[2]$, found a few more samples, then got tired and asked a computer to carry on.

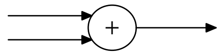
Block diagrams, the third representation, seem to imply a sample-by-sample analysis because the delay element holds on to samples, spitting out the sample after one time tick. But block diagrams live in both worlds and can also represent operations on whole signals. Just reinterpret the elements in the whole-signal view, as follows:



action 1: multiply whole signal by α



action 2: shift whole signal right one tick



combinator: add whole signals

To benefit from the abstraction, compactly represent the preceding three elements. When a signal is a single object, the gain element acts like ordinary multiplication, and the plus element acts like addition of numbers. If the delay element could also act like an arithmetic operation, then all three elements would act in a familiar way, and block diagrams could be manipulated using the ordinary rules of algebra. In order to bring the delay element into this familiar framework, we introduce the operator representation.

3.4.1 Operator representation

In operator notation, the symbol $\mathcal{R}$ stands for the right-shift operator. It takes a signal and shifts it one step to the right. Here is the notation for a system that delays a signal X by one tick to produce a signal Y:

$$Y = \mathcal{R}\{X\}.$$

Now forget the curly braces, to simplify the notation and to strengthen the parallel with ordinary multiplication. The clean notation is

$$Y = \mathcal{R}X.$$

Pause to try 21. Convince yourself that right-shift operator $\mathcal{R}$, rather than the left-shift operator $\mathcal{L}$, is equivalent to a delay.

Let's test the effect of applying $\mathcal{R}$ to the fundamental signal, the impulse. The impulse is

$$I = 1, 0, 0, 0, \ldots$$

Applying $\mathcal{R}$ to it gives

$$\mathcal{R}I = 0, 1, 0, 0, \ldots$$

which is also the result of delaying the signal by one time tick. So $\mathcal{R}$ rather than $\mathcal{L}$ represents the delay operation. In operator notation, the block-diagram elements are:

α	action 1 (gain)	multiply whole signal by α
$\mathcal{R}$	action 2 (delay)	shift whole signal right one tick
$+$	combinator	add whole signals

3.4.2 Using operators to rewrite difference equations

Let's try operator notation on the first example of the chapter: rewriting the single difference equation

$$d[n] = a[n] - 3a[n-1] + 3a[n-2] - a[n-3]$$

into the system of three difference equations

$$\begin{aligned} d[n] &= c[n] - c[n-1], \\ c[n] &= b[n] - b[n-1], \\ b[n] &= a[n] - a[n-1]. \end{aligned}$$

To turn the sample-by-sample notation into whole-signal notation, turn the left side of the long equation into the whole signal D, composed of the samples $d[0], d[1], d[2], \ldots$. Turn the samples on the right side into whole signals as follows:

$$\begin{aligned} a[n] &\to A, \\ a[n-1] &\to \mathcal{R}A, \\ a[n-2] &\to \mathcal{R}\mathcal{R}A, \\ a[n-3] &\to \mathcal{R}\mathcal{R}\mathcal{R}A. \end{aligned}$$

Now import compact notation from algebra: If $\mathcal{R}$ acts like a variable or number then $\mathcal{R}\mathcal{R}$ can be written $\mathcal{R}^2$. Using exponent notation, the translations are:

$$\begin{aligned} a[n] &\to A, \\ a[n-1] &\to \mathcal{R}A, \\ a[n-2] &\to \mathcal{R}^2A, \\ a[n-3] &\to \mathcal{R}^3A. \end{aligned}$$

With these mappings, the difference equation turns into the compact form

$$D = (1 - 3\mathcal{R} + 3\mathcal{R}^2 - \mathcal{R}^3)A.$$

To show that this form is equivalent to the system of three difference equations, translate them into an operator expression connecting the input signal A and the output signal D.

Pause to try 22. What are the operator versions of the three difference equations?

The system of equations turns into these operator expressions

$$\begin{aligned} d[n] &= c[n] - c[n-1] &&\rightarrow && D = (1-\mathcal{R})C, \\ c[n] &= b[n] - b[n-1] &&\rightarrow && C = (1-\mathcal{R})B, \\ b[n] &= a[n] - a[n-1] &&\rightarrow && B = (1-\mathcal{R})A. \end{aligned}$$

Eliminate B and C to get

$$D = (1-\mathcal{R})(1-\mathcal{R})(1-\mathcal{R})A = (1-\mathcal{R})^3 A.$$

Expanding the product gives

$$D = (1 - 3\mathcal{R} + 3\mathcal{R}^2 - \mathcal{R}^3)A,$$

which matches the operator expression corresponding to the single difference equation. The operator derivation of the equivalence is simpler than the block-diagram rewriting, and much simpler than the difference-equation manipulation.

Now extend the abstraction by dividing out the input signal:

$$\frac{D}{A} = 1 - 3\mathcal{R} + 3\mathcal{R}^2 - \mathcal{R}^3.$$

The operator expression on the right, being independent of the input and output signals, is a characteristic of the system alone and is called the **system functional**.

The moral of the example is that operators help you efficiently analyze systems. They provide a grammar for combining, for subdividing, and in general for rewriting systems. It is a familiar grammar, the grammar of algebraic expressions. Let's see how extensively operators follow these. In the next section we stretch the analogy and find that it does not break.

Exercise 25. What is the result of applying $1 - \mathcal{R}$ to the signal $1, 2, 3, 4, 5, \ldots$?

Exercise 26. What is the result of applying $(1 - \mathcal{R})^2$ to the signal $1, 4, 9, 16, 25, 36, \ldots$?

3.5 Feedback connections

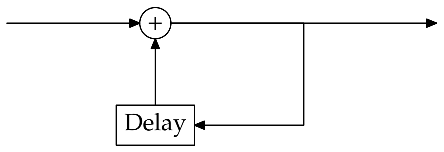
The system with $(1-\mathcal{R})^3$ as its system functional used only feedforward connections: The output could be computed directly from a fixed number of inputs. However, many systems – such as Fibonacci or bank accounts – contain feedback, where the output depends on previous values of the output. Feedback produces new kinds of system functionals. Let's test whether they also obey the rules of algebra.

3.5.1 Accumulator

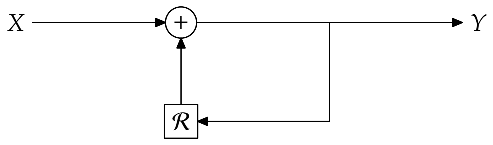
Here is the difference equation for the simplest feedback system, an **accumulator**:

$$y[n] = y[n-1] + x[n].$$

It is a bank account that pays no interest. The output signal (the balance) is the sum of the inputs (the deposits, whether positive or negative) up to and including that time. The system has this block diagram:



Now combine the visual virtues of block diagrams with the compactness and symbolic virtues of operators by using $\mathcal{R}$ instead of 'Delay'. The operator block diagram is



Pause to try 23. What is its system functional?

Either from this diagram or from the difference equation, translate into operator notation:

$$Y = \mathcal{R}Y + X.$$

Collect the Y terms on one side, and you find end up with the system functional:

$$\frac{Y}{X} = \frac{1}{1-\mathcal{R}}.$$

It is the reciprocal of the differentiator.

This operator expression is the first to include $\mathcal{R}$ in the denominator. One way to interpret division is to compare the output signal produced by the difference equation with the output signal produced by the system functional $1/(1-\mathcal{R})$. For simplicity, test the equivalence using the impulse

$$I = 1, 0, 0, 0, \ldots$$

as the input signal. So $x[n]$ is 1 for $n = 0$ and is 0 otherwise. Then the difference equation

$$y[n] = y[n-1] + x[n]$$

produces the output signal

$$Y = 1, 1, 1, 1, \ldots.$$

Exercise 27. Check this claim.

The output signal is the discrete-time **step function** θ. Now apply $1/(1-\mathcal{R})$ to the impulse I by importing techniques from algebra or calculus. Use synthetic division, Taylor series, or the binomial theorem to rewrite $1/(1-\mathcal{R})$ as

$$\frac{1}{1-\mathcal{R}} = 1 + \mathcal{R} + \mathcal{R}^2 + \mathcal{R}^3 + \cdots.$$

To apply $1/(1-\mathcal{R})$ to the impulse, apply the each of terms $1, \mathcal{R}, \mathcal{R}^2, \ldots$ to the impulse I:

$$\begin{aligned}
1I &= 1,0,0,0,0,0,0,\ldots,\\
\mathcal{R}I &= 0,1,0,0,0,0,0,\ldots,\\
\mathcal{R}^2I &= 0,0,1,0,0,0,0,\ldots,\\
\mathcal{R}^3I &= 0,0,0,1,0,0,0,\ldots,\\
\mathcal{R}^4I &= 0,0,0,0,1,0,0,\ldots,\\
&\ldots
\end{aligned}$$

Add these signals to get the output signal Y.

Pause to try 24. What is Y?

For $n \geq 0$, the $y[n]$ sample gets a 1 from the $\mathcal{R}^n I$ term, and from no other term. So the output signal is all 1's from $n = 0$ onwards. The signal with those samples is the step function:

$$Y = 1,1,1,1,\ldots.$$

Fortunately, this output signal matches the output signal from running the difference equation. So, for an impulse input signal, these operator expressions are equivalent:

$$\frac{1}{1-\mathcal{R}} \quad \text{and} \quad 1+\mathcal{R}+\mathcal{R}^2+\mathcal{R}^3+\cdots.$$

Exercise 28. If you are mathematically inclined, convince yourself that verifying the equivalence for the impulse is sufficient. In other words, we do not need to try all other input signals.

The moral is that the $\mathcal{R}$ operator follows the rules of algebra and calculus. So have courage: Use operators to find results, and do not worry.

3.5.2 Fibonacci

Taking our own advice, we now analyze the Fibonacci system using operators. The recurrence is:

output = delayed output + twice-delayed output + input.

Pause to try 25. Turn this expression into a system functional.

The output signal is F, and the input signal is X. The delayed output is $\mathcal{R}X$, and the twice-delayed output is $\mathcal{R}\mathcal{R}X$ or $\mathcal{R}^2X$. So

$$F = \mathcal{R}F + \mathcal{R}^2F + X.$$

Collect all F terms on one side:

$$F - \mathcal{R}F - \mathcal{R}^2F = X.$$

Then factor the F:

$$(1 - \mathcal{R} - \mathcal{R}^2)F = X.$$

Then divide both sides by the $\mathcal{R}$ expression:

$$F = \frac{1}{1 - \mathcal{R} - \mathcal{R}^2}X.$$

So the system functional is

$$\frac{1}{1 - \mathcal{R} - \mathcal{R}^2}.$$

Exercise 29. Using `maxima` or otherwise, find the Taylor series for $1/(1 - \mathcal{R} - \mathcal{R}^2)$. What do you notice about the coefficients? Coincidence? `maxima` (**maxima.sourceforge.net**) is a powerful symbolic algebra and calculus system. It descends from `Macsyma`, which was created at MIT in the 1960's. `maxima` is free software (GPL license) and is available for most platforms.

3.6 Summary

Including the two system representations discussed in this chapter, you have four representation for discrete-time systems:

1. verbal descriptions,
2. difference equations,
3. block diagrams, and
4. operator expressions.

In the next chapter, we use the operator representation to decompose, and design systems.

Recitation 4
Modes

The goals of this chapter are:

- to illustrate the experimental way that an engineer studies systems, even abstract, mathematical systems;
- to illustrate what modes are by finding them for the Fibonacci system; and
- to decompose second-order systems into modes, explaining the decomposition using operators and block diagrams.

The first question is what a mode is. That question will be answered as we decompose the Fibonacci sequence into simpler sequences. Each simple sequence can be generated by a first-order system like the leaky tank and is called a **mode** of the system. By decomposing the Fibonacci sequence into modes, we decompose the system into simpler, first-order subsystems.

The plan of the chapter is to treat the Fibonacci system first as a black box producing an output signal F and to develop computational probes to examine signals. This experimental approach is how an engineer studies even abstract, mathematical systems. The results from the probes will show us how to decompose the signal into its modes. These modes are then reconciled with what the operator method predicts for decomposing the system.

Why describe the experimental, and perhaps harder, method for finding the modes before giving the shortcuts using operators? We know the operator expression for the Fibonacci system, and could just rewrite it using algebra. The answer is that the operator method has meaning only after you feel modes in your fingertips, a feeling developed only as you play with signals. Without first playing, we would be teaching you amazing feats of calculation on meaningless objects.

Furthermore, the experimental approach works even when no difference equation is available to generate the sequence. Engineers often characterize such unknown or partially known systems. The system might be:

- *computational:* Imagine debugging someone else's program. You send in test inputs to find out how it works and what makes it fail.
- *electronic:* Imagine debugging a CPU that just returned from the fabrication run, perhaps in quantities of millions, but that does not correctly divide floating-point numbers [9]. You might give it numbers to divide until you find the simplest examples that give wrong answers. From that data you can often deduce the flaw in the wiring.
- *mathematical:* Imagine computing primes to investigate the twin-prime conjecture [10], one of the outstanding unsolved problems of number theory. [The conjecture states that there are an infinite number of prime pairs $p, p+2$, such as $(3,5)$, $(5,7)$, etc.] The new field of experimental mathematics, which uses computational tools to investigate mathematics problems, is lively, growing, and a fertile field for skilled engineers [11, 12, 13].

So we hope that, through experimental probes of the Fibonacci sequence, you learn a general approach to solving problems.

4.1 Growth of the Fibonacci series

Section 2.3.3 estimated how fast the sequence $f[n]$ grew. It seemed to grow geometrically with an order of growth between $\sqrt{2}$ and 2. Our first project is to experimentally narrow this range and thereby to guess a closed form for the order of growth.

One probe to find the order of growth is to compute the successive ratios $f[n]/f[n-1]$. The ratios oscillated around 1.618, but this estimate is not accurate enough to guess a closed form. Since the oscillations in the ratio die out as n grows, let's estimate the ratio accurately by computing it for large n. Our tool for these experiments – our probe – is a computer that we program in Python, a clean, widely available language. Use any tool that fits you, perhaps another language, a graphing calculator, or a spreadsheet.

Section 2.3.2 offered this Python code to compute $f[n]$:

```
def f(n):
  if n < 2: return 1
  return f(n-1) + f(n-2)
```

But the code is slow when n is large. Here are the running times to evaluate $f[n]$ on a Pentium CoreDuo 1.8GHz processor:

n	10	15	20	25	30
time (ms)	0.17	1.5	21	162	1164

The times grow rapidly!

Exercise 30. What is the running time of this implementation?

The times might seem low enough to be usable, but imagine being on a desert island with only a graphing calculator; then the times might be a factor of 10 or of 100 longer. We would like to build an efficient computational probe so that it is widely usable.

An efficient function would store previously computed answers, returning the stored answer when possible rather than recomputing old values. In Python, one can store the values in a dictionary, which is analogous to a hash in Perl or an associative array in `awk`. The memoized version of the Fibonacci function is:

```
memo = {}
def f(n):
  if n < 2      : return 1
  if n in memo : return memo[n]
  memo[n] = f(n-1) + f(n-2)
  return memo[n]
```

Pause to try 26. What is the running time of the memoized function, in big-Oh notation?

The new function runs in linear time – a faster probe! – so we can inexpensively compute $f[n]$ for large n. Here are the ratios $f[n]/f[n-1]$:

n	$f[n]/f[n-1]$
5	1.60000000000000009
10	1.61818181818181817
15	1.61803278688524599
20	1.61803399852180330
25	1.61803398867044312
30	1.61803398875054083
35	1.61803398874988957
40	1.61803398874989490
45	1.61803398874989490

These values are very stable by $n = 45$, perhaps limited in stability only by the precision of the floating-point numbers.

Let's see what closed form would produce the ratio 1.61803398874989490 at $n = 45$. One source for closed forms is your intuition and experience. Another wonderful source is the Inverse Symbolic Calculator **http://oldweb.cecm.sfu.ca/projects/ISC/ISCmain.html**. By using the Inverse Symbolic Calculator, you increase your repertoire of closed form and thereby enhance your intuition.

Pause to try 27. Ask the Inverse Symbolic Calculator about 1.61803398874989490.

The Inverse Symbolic Calculator thinks that 1.61803398874989490 is most likely the positive root of $x^2 - x - 1$ or, equivalently, is the golden ratio ϕ:

$$\phi \equiv \frac{1+\sqrt{5}}{2}$$

Let's use that hypothesis. Then

$$f[n] \sim \phi^n.$$

But we do not know the constant hidden by the $\sim$ symbol. Find that constant by using the Inverse Symbolic Calculator one more time. Here is a table of the ratio $f[n]/\phi^n$. With luck it converges to a constant. And it does:

n	$f[n]/\phi^n$
0	1.00000000000000000
10	0.72362506926471781
20	0.72360679895785285
30	0.72360679775005809
40	0.72360679774997805
50	0.72360679774997783
60	0.72360679774997749
70	0.72360679774997727
80	0.72360679774997705
90	0.72360679774997672
100	0.72360679774997649

Around $n = 10$, the ratios look like $\sqrt{3} - 1 \approx 0.732$ but later ratios stabilize around a value inconsistent with that guess.

Pause to try 28. Ask the Inverse Symbolic Calculator about 0.72360679774997649. Which of the alternatives seem most reasonable?

The Inverse Symbolic Calculator provides many closed forms for 0.72360679774997649. A choice that contains $\sqrt{5}$ is reasonable since ϕ contains $\sqrt{5}$. The closed form nearest to 0.72360679774997649 and containing $\sqrt{5}$ is $(1 + 1/\sqrt{5})/2$, which is also $\phi/\sqrt{5}$. So the Fibonacci sequence is roughly

$$f[n] \approx \frac{\phi}{\sqrt{5}}\phi^n.$$

4.2 Taking out the big part from Fibonacci

Now let's **take out the big part** by peeling away the $\frac{\phi}{\sqrt{5}}\phi^n$ contribution to see what remains. Define the signal F_1 by

$$f_1[n] = \frac{\phi}{\sqrt{5}}\phi^n.$$

This signal is one mode of the Fibonacci sequence. The shape of a mode is its order of growth, which here is ϕ. The amplitude of a mode is the prefactor, which here is $\phi/\sqrt{5}$. The mode shape is a characteristic of the system, whereas the amplitude depends on the input signal (for this example, the input signal was the impulse). So we often have more interest in the shape than in the amplitude. However, here we need shape and amplitude in order to determine the signal and peel it away.

So tabulate the residual signal $F_2 = F - F_1$:

n	$f_2[n] = f[n] - f_1[n]$
0	+0.27639320225002106
1	−0.17082039324993681
2	+0.10557280900008426
3	−0.06524758424985277
4	+0.04032522475023104
5	−0.02492235949962307
6	+0.01540286525060708
7	−0.00951949424901599
8	+0.00588337100158753
9	−0.00363612324743201
10	+0.00224724775415552

The residual signal starts small and gets smaller, so the main mode F_1 is an excellent approximation to the Fibonacci sequence F. To find a closed form for the residual signal F_2, retry the successive-ratios probe:

n	$f_2[n]/f_2[n-1]$
1	−0.61803398874989446
2	−0.61803398874989601
3	−0.61803398874989390
4	−0.61803398874989046
5	−0.61803398874993953

6	−0.61803398874974236
7	−0.61803398875029414
8	−0.61803398874847626
9	−0.61803398875421256
10	−0.61803398873859083

The successive ratios are almost constant and look suspiciously like $1 - \phi$, which is also $-1/\phi$.

Exercise 31. Show that $1 - \phi = -1/\phi$.

So $f_2[n] \sim (-\phi)^{-n}$. To evaluate the amplitude, divide $f_2[n]$ by the mode shape $(-\phi)^{-n}$. Here is a table of those results:

n	$f_2[n]/(-\phi)^{-n}$
1	0.27639320225002090
2	0.27639320225002140
3	0.27639320225002101
4	0.27639320225001901
5	0.27639320225003899
6	0.27639320224997083
7	0.27639320225014941
8	0.27639320224951497
9	0.27639320225144598
10	0.27639320224639063

Those values stabilize quickly and look like one minus the amplitude of the ϕ^n mode. So the amplitude of the $(-\phi)^n$ mode is $1 - \phi/\sqrt{5}$, which is also $1/(\phi\sqrt{5})$. Thus the residual signal, combining its shape and amplitude, is

$$f_2[n] = \frac{1}{\phi\sqrt{5}}(-\phi)^{-n}.$$

Now combine the F_1 and F_2 signals to get the Fibonacci signal:

$$\begin{aligned} f[n] &= f_1[n] + f_2[n] \\ &= \frac{\phi}{\sqrt{5}}\phi^n + \frac{1}{\phi\sqrt{5}}(-\phi)^{-n}. \end{aligned}$$

This closed form, deduced using experiment, is the famous Binet formula for the n^{th} Fibonacci number.

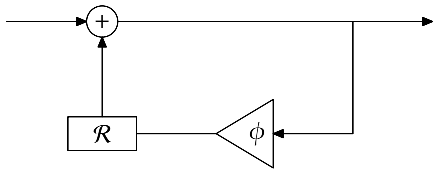
Exercise 32. Use peeling away and educated guessing to find a closed form for the output signal when the impulse is fed into the following difference equation:

$$y[n] = 7y[n-1] - 12y[n-2] + x[n].$$

4.3 Operator interpretation

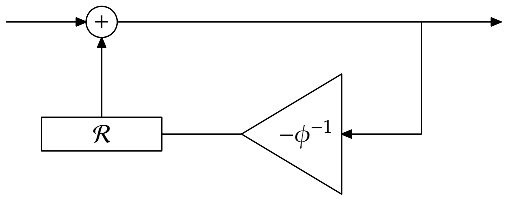
Next we interpret this experimental result using operators and block diagrams. Modes are the simplest persistent responses that a system can make, and are the building blocks of all systems, so we would like to find the operator or block-diagram representations for a mode.

The Fibonacci signal decomposed into two simpler signals F_1 and F_2 – which are also the modes – and each mode grows geometrically. Geometric growth results from one feedback loop. So the ϕ^n mode is produced by this system



with the system functional $(1-\phi\mathcal{R})^{-1}$.

The $(-\phi)^{-n}$ mode is produced by this system



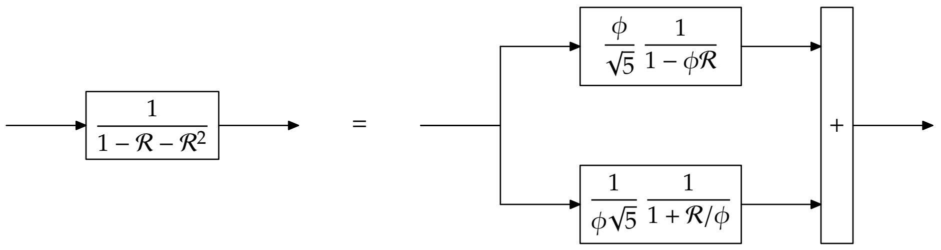
with the system functional $(1+\mathcal{R}/\phi)^{-1}$.

The Fibonacci system is the sum of these signals scaled by the respective amplitudes, so its block diagram is a weighted sum of the preceding block diagrams. The system functional for the Fibonacci system is a weighted sum of the pure-mode system functionals.

So let's add the individual system functionals and see what turns up:

$$\begin{aligned} F(\mathcal{R}) &= F_1(\mathcal{R}) + F_2(\mathcal{R}) \\ &= \frac{\phi}{\sqrt{5}}\frac{1}{1-\phi\mathcal{R}} + \frac{1}{\phi\sqrt{5}}\frac{1}{1+\mathcal{R}/\phi} \\ &= \frac{1}{1-\mathcal{R}-\mathcal{R}^2}. \end{aligned}$$

That functional is the system functional for the Fibonacci system derived directly from the block diagram (**Section 3.5.2**)! So the experimental and operator approaches agree that these operator block diagrams are equivalent:


$\frac{1}{1-\mathcal{R}-\mathcal{R}^2}$
=
$\frac{\phi}{\sqrt{5}}\frac{1}{1-\phi\mathcal{R}}$
$\frac{1}{\phi\sqrt{5}}\frac{1}{1+\mathcal{R}/\phi}$
+

where, to make the diagram easier to parse, system functionals stand for the first- and second-order systems that they represent.

Exercise 33. Write the system of difference equations that corresponds to the parallel-decomposition block diagram. Show that the system is equivalent to the usual difference equation

$$f[n] = f[n-1] + f[n-2] + x[n].$$

The equivalence is obvious neither from the block diagrams nor from the difference equations directly. Making the equivalence obvious needs either experiment or the operator representation. Having experimented, you are ready to use the operator representation generally to find modes.

4.4 General method: Partial fractions

So we would like a way to decompose a system without peeling away and guessing. And we have one: the method of partial fractions, which shows the value of the operator representation and system functional. Because the system functional behaves like an algebraic expression – or one might say, because it is an algebraic expression – it is often easier to manipulate than is the block diagram or the difference equation.

Having gone from the decomposed first-order systems to the original second-order system functional, let's now go the other way: from the original system functional to the decomposed systems. To do so, first factor the $\mathcal{R}$ expression:

$$\frac{1}{1-\mathcal{R}-\mathcal{R}^2} = \frac{1}{1-\phi\mathcal{R}}\,\frac{1}{1+\mathcal{R}/\phi}.$$

This factoring, a series decomposition, will help us study poles and zeros in a later chapter. Here we use it to find the parallel decomposition by using the technique of partial fractions.

The partial fractions should use the two factors in denominator, so guess this form:

$$\frac{1}{1-\mathcal{R}-\mathcal{R}^2} = \frac{a}{1-\phi\mathcal{R}} + \frac{b}{1+\mathcal{R}/\phi},$$

where a and b are unknown constants. After adding the fractions, the denominator will be the product $(1-\phi\mathcal{R})(1+\mathcal{R}/\phi)$ and the numerator will be the result of cross multiplying:

$$a(1+\mathcal{R}/\phi) + b(1-\phi\mathcal{R}) = a + (a/\phi)\mathcal{R} + b - b\phi\mathcal{R}.$$

We want the numerator to be 1. If we set $a = \phi$ and $b = 1/\phi$, then at least the $\mathcal{R}$ terms cancel, leaving only the constant $a + b$. So we chose a and b too large by the sum $a + b$, which is $\phi + 1/\phi$ or $\sqrt{5}$. So instead choose

$$a = \phi/\sqrt{5},$$
$$b = 1/(\phi\sqrt{5}).$$

If you prefer solving linear equations to the guess-and-check method, here are the linear equations:

$$a + b = 1,$$
$$a/\phi - b\phi = 0,$$

whose solutions are the ones deduced using the guess-and-check method.

The moral: To find how a system behaves, factor its system functional and use partial fractions to decompose that factored form into a sum of first-order systems. With that decomposition, you can predict the output signal because you know how first-order systems behave.

You can practice the new skill of decomposition with the following question:

Exercise 34. Look again at the system

$$y[n] = 7y[n-1] - 12y[n-2] + x[n].$$

Decompose the operator representation into a sum of two modes and draw the corresponding block diagram (using block diagram elements). When the input signal X is the impulse, do the operator and block-diagram decompositions produce the same closed form that you find by peeling away and guessing?

Recitation 5
Repeated roots

After reading this chapter you should be able

- to use a continuity argument
- to explain the non-geometric output of a mode with a double root.

Modes generate persistent outputs. So far our examples generate persistent geometric sequences. But a mode from a repeated root, such as from the system functional $(1 - \mathcal{R}/2)^{-3}$, produces outputs that are not geometric sequences. How does root repetition produce this seemingly strange behavior?

The analysis depends on the idea that repeated roots are an unlikely, special situation. If roots scatter randomly on the complex plane, the probability is zero that two roots land exactly on the same place. A generic, decent system does not have repeated roots, and only through special contrivance does a physical system acquire repeated roots. This fact suggests deforming a repeated-root system into a generic system by slightly moving one root so that the modes of the deformed system produce geometric sequences. This new system is therefore qualitatively easier to analyze than is the original system, and it can approximate the original system as closely as desired. This **continuity argument** depends on the idea that the world changes smoothly: that a small change to a system, such as by moving a root a tiny amount, produces only a small change to the system's output.

To generate a double root, we use the *RC* circuit (**Section 1.1.2**) or the leaky tank (**Section 1.1**). Either system alone has one mode. To make a double root, cascade two identical tanks or *RC* circuits, where the output of the first system is the input to the second system.

Exercise 35. When making an *RC* cascade system analogous to a cascade of two leaky tanks, does the *RC* cascade need a unity-gain buffer between the two *RC* circuits?

5.1 Leaky-tank background

Let the leaky tank or *RC* circuit have time constant τ. Let V_{in} be the input signal, which is the flow rate into the tank system or the input voltage in the *RC* circuit, and let V_{out} be the output signal. Then the differential equation of either system is

$$\tau \dot{V}_{\text{out}} = V_{\text{in}} - V_{\text{out}}.$$

Convert this equation into a discrete-time system using the forward-Euler approximation (**Section 1.3**). Using a time step T, the difference equation becomes

$$\tau \frac{V_{\mathrm{out}}[n] - V_{\mathrm{out}}[n-1]}{T} = V_{\mathrm{in}}[n-1] - V_{\mathrm{out}}[n-1].$$

To promote equation hygiene, define the dimensionless ratio $\epsilon \equiv T/\tau$ and collect like terms. The clean difference equation is

$$V_{\mathrm{out}}[n] - (1-\epsilon)V_{\mathrm{out}}[n-1] = \epsilon V_{\mathrm{in}}[n-1].$$

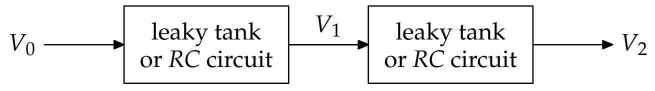
Pause to try 29. What system functional corresponds to this difference equation?

The corresponding system functional is

$$\frac{V_{\mathrm{out}}}{V_{\mathrm{in}}} = \frac{\epsilon\mathcal{R}}{1-(1-\epsilon)\mathcal{R}}.$$

Exercise 36. What block diagram corresponds to this system functional?

A double root arises from cascading two identical systems. Here is its high-level block diagram showing the input, intermediate, and output signals:



Its system functional is the square of the functional for one system:

$$\frac{V_2}{V_0} = \left(\frac{\epsilon\mathcal{R}}{1-(1-\epsilon)\mathcal{R}}\right)^2.$$

The numerator $(\epsilon\mathcal{R})^2$ does not add interesting features to the analysis, so simplify life by ignoring it. To simplify the algebra further, define $\beta = 1-\epsilon$. With that definition and without the boring $\epsilon\mathcal{R}$ factor, the purified system is

$$\frac{V_2}{V_0} = \frac{1}{(1-\beta\mathcal{R})^2}.$$

5.2 Numerical computation

By design, this cascade system has a double root at $\beta = 1-\epsilon$. Let's simulate its impulse response and find patterns in the data. Simulation requires choosing numerical values for the parameters, and here the only parameter is $\epsilon = T/\tau$. An accurate discretization uses a time step T much shorter than the system time constant τ; otherwise the system changes significantly between samples, obviating the discrete-time approximation. So use $\epsilon \ll 1$.

Pause to try 30. Write a program to simulate the impulse response, choosing a reasonable ϵ or β.

The following simulation uses $\epsilon = 0.05$ or $\beta = 0.95$ in computing the impulse response:

```
from scipy import *

N = 100
impulse = zeros(N)
impulse[0] = 1
beta = 0.95

# return the output signal from feeding INPUT signal through a system
# with a feedback loop containing a delay and the given GAIN.
def onestage(input, gain):
    output = input * 0
    output[0] = input[0]
    for i in range(1,len(output)):      # 1..n-1
        output[i] = input[i] + gain*output[i-1]
    return output

signal = impulse                    # start with the impulse
for gain in [beta, beta]:           # run it through each system
    signal = onestage(signal, gain)
print signal
```

The impulse response is:

n	$y[n]$
0	1.000000
1	1.900000
2	2.707500
3	3.429500
4	4.072531
5	4.642686
6	5.145643
7	5.586698
8	5.970784
9	6.302494
10	6.586106
...	

5.3 Analyzing the output signal

The impulse response contains a pattern. To find it, play with the data and make conjectures. The first few samples look like $n + 1$. However, by $n = 10$ that conjecture looks dubious. So look for another pattern. A single system $(1 - \beta\mathcal{R})^{-1}$ would have a geometric-sequence output where the nth sample is β^n. Maybe that geometric decay appears in the double system and swamps the conjectured $n+1$ growth. Therefore, take out the big part from the impulse response by tabulating the signal $y[n]/0.95^n$. To do so, add one line of code to the previous program:

```
print signal/0.95**arange(N)
```

The data are

n	$y[n]/0.95^n$
0	1.000000
1	2.000000
2	3.000000
3	4.000000
4	5.000000
5	6.000000
6	7.000000
7	8.000000
8	9.000000
9	10.000000
10	11.000000

Now $y[n] = n + 1$ is exact! The impulse response of the double cascade is the signal

$$y[n] = (n+1) \cdot 0.95^n \qquad \text{for } n \geq 0.$$

The factor of 0.95^n makes sense because a single system $(1 - 0.95\mathcal{R})^{-1}$ would have 0.95^n as its impulse response. But how does the factor of $n + 1$ arise? To understand its origin, one method is convolution, which was discussed in the lecture. Here we show an alternative method using a continuity argument.

5.4 Deforming the system: The continuity argument

The cascade is hard to analyze because its roots are replicated. So deform the cascade by making the second root be 0.951 instead of 0.95. That slightly deformed system has the functional

$$\frac{1}{1 - 0.951\mathcal{R}} \cdot \frac{1}{1 - 0.95\mathcal{R}}.$$

Since the root hardly moved, the impulse response should be almost the same as the impulse response of the original system. This assumption is the essence of the continuity argument. We could find the response by slightly modifying the preceding program. However, reaching for a program too often does not add insight.

Alternatively, now that the system's roots are unequal, we can easily use partial fractions. The first step in partial fractions is to find the modes:

$$M_1 = \frac{1}{1 - 0.951\mathcal{R}} \qquad \text{and} \qquad M_2 = \frac{1}{1 - 0.95\mathcal{R}}.$$

The system functional is a linear combination of these modes:

$$\frac{1}{1 - 0.951\mathcal{R}} \cdot \frac{1}{1 - 0.95\mathcal{R}} = \frac{C_1}{1 - 0.951\mathcal{R}} - \frac{C_2}{1 - 0.95\mathcal{R}}.$$

Exercise 37. Show that $C_1 = 951$ and $C_2 = 950$.

The partial-fractions decomposition is

$$\frac{1}{1-0.95\mathcal{R}} \cdot \frac{1}{1-0.951\mathcal{R}} = \frac{1}{0.001}\left(\frac{0.951}{1-0.951\mathcal{R}} - \frac{0.95}{1-0.95\mathcal{R}}\right).$$

The $0.951/(1-0.951\mathcal{R})$ system contributes the impulse response 0.951^{n+1}, and the $0.95/(1-0.95\mathcal{R})$ system contributes the impulse response 0.95^{n+1}.

Exercise 38. Check these impulse responses.

So the impulse response of the deformed system is

$$y[n] = 1000 \cdot (0.951^{n+1} - 0.95^{n+1}).$$

Since $0.951 \approx 0.95$, the difference in parentheses is tiny. However, the difference is magnified by the factor of 1000 outside the parentheses. The resulting signal is not tiny, and might contain the non-geometric factor of $n+1$ in the impulse response of a true double root.

To approximate the difference $0.951^{n+1} - 0.95^{n+1}$, use the binomial theorem, keeping only the two largest terms:

$$\begin{aligned} 0.951^{n+1} &= (0.95 + 0.001)^{n+1} \\ &\approx 0.95^{n+1} + (n+1)0.95^{n} \cdot 0.001 + \cdots. \end{aligned}$$

Thus the approximate impulse response is

$$y[n] \approx 1000 \cdot (n+1) \cdot 0.95^{n} \cdot 0.001.$$

The factor of 1000 cancels the factor of 0.001 to leave

$$y[n] \approx (n+1) \cdot 0.95^{n},$$

which is what we conjectured numerically!

Thus the linear prefactor $n+1$ comes from subtracting two garden-variety, geometric-sequence modes that are almost identical. The $\approx$ sign reflects that we kept only the first two terms in the binomial expansion of 0.951^{n+1}. However, as the deformation shrinks, the shifted root at 0.951 becomes instead 0.9501 or 0.95001, etc. As the root approaches 0.95, the binomial approximation becomes exact, as does the impulse response $(n+1) \cdot 0.95^{n}$.

The response $(n+1) \cdot 0.95^{n}$ is the product of an increasing function with a decreasing function, with each function fighting for victory. In such situations, one function usually wins at the $n \to 0$ extreme, and the other function wins at the $n \to \infty$ extreme, with a maximum product where the two functions arrange a draw.

Exercise 39. Sketch $n+1$, 0.95^{n}, and their product.

Pause to try 31. Where is the maximum of $(n+1)\cdot 0.95^n$?

This product reaches a maximum when two successive samples are equal. The ratio of successive samples is

$$\frac{y[n]}{y[n-1]} = \frac{0.95^n \cdot (n+1)}{0.95^{n-1}\cdot n} = 0.95\cdot\left(1+\frac{}{n}\right).$$

That ratio is one when $n = 19$. So $y[19]$ and $y[18]$ are the maximums of the impulse response. The signal rises till then, plateaus, and then decays to zero.

5.5 Higher-order cascades

The propagation of an impulse in a neuronal dendrite – which has active amplification and regeneration – is a continuous-space *RC* cascade. It can be simulated approximately as a giant cascade of discrete-space *RC* filters.

Rather than try a 1000-element cascade, try the impulse response of a triple cascade. Guess before calculating, whether calculating numerically or analytically. The single system $(1-\beta\mathcal{R})^{-1}$ produces plain geometric decay β^n with no prefactor. The double system $(1-\beta\mathcal{R})^{-2}$ produces geometric decay with a linear prefactor $n+1$. So a triple cascade should produce geometric decay with a quadratic prefactor. And it does.

Exercise 40. Guess the quadratic prefactor of the impulse response of the triple cascade. How can you confirm your guess?

Exercise 41. Compute and sketch the output signal of the triple cascade.

Exercise 42. Where is the maximum of the impulse response? How does it compare to the maximum for the double cascade?

Recitation 6
The perfect (sine) wave

The goals of this chapter are:

- to analyze several methods for discretizing a continuous-time system; and
- to illustrate complex poles and the significance of the unit circle.

How can you compute a sine wave if your programming language did not have a built-in sine function? You can use the coupled oscillator from the first problem set:

$$\begin{aligned}\dot{y}_1 &= y_2,\\ \dot{y}_2 &= -y_1.\end{aligned}$$

Let's rewrite the equations to have physical meaning. Imagine y_1 as the oscillator's position x. Then y_2 is $\dot{x}$, which is the oscillator's velocity. So replace y_1 with x, and replace y_2 with v. Then $\dot{y}_2$ is the acceleration a, making the equations

$$\begin{aligned}\dot{x} &= v,\\ a &= -x.\end{aligned}$$

The first equation is a purely mathematical definition, so it has no physical content. But the second equation describes the acceleration of an ideal spring with unit spring constant and unit mass. So the position $x(t)$ is a linear combination of $\sin t$ and $\cos t$. An accurate discrete-time simulation, if we can make one, would reproduce these sinusoidal oscillations, and this chapter shows you how.

To turn the system into a discrete-time system, let T be the time step, $x[n]$ be the discrete-time estimate for the position $x(nT)$, and $v[n]$ be the discrete-time estimate for the velocity $v(nT)$. Those changes take care of all the terms except for the derivatives. How do you translate the derivatives? Three methods are easy to use, and we try each, finding its poles and analyzing its fidelity to the original, continuous-time system.

6.1 Forward Euler

The first method for translating the derivatives is the forward-Euler approximation. It estimates the continuous-time derivatives $\dot{x}(nT)$ and $\dot{v}(nT)$ using the forward differences

$$\begin{aligned}\dot{x}(nT) &\rightarrow \frac{x[n+1]-x[n]}{T},\\ \dot{v}(nT) &\rightarrow \frac{v[n+1]-v[n]}{T}.\end{aligned}$$

Then the continuous-time sysytem becomes

$$x[n+1] - x[n] = Tv[n],$$
$$v[n+1] - v[n] = -Tx[n].$$

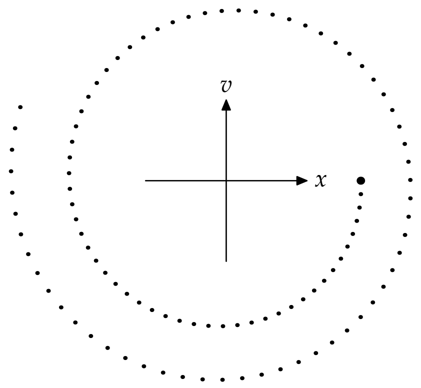
The new samples $x[n+1]$ and $v[n+1]$ depend only on the old samples $x[n]$ and $v[n]$. So this system provides an explicit recipe for computing later samples.

6.1.1 Simulation

Here is Python code to implement these equations. It starts the system with the initial conditions $x[0] = 1$ and $v[0] = 0$ and plots v vs x.

```
from scipy import *
import pylab as p

T = 0.1

N = int(T**-2)
x = zeros(N)
x[0] = 1
v = zeros(N)

for n in range(N-1):
    x[n+1] = x[n] + T*v[n]
    v[n+1] = v[n] - T*x[n]

p.plot(x,v,'r.')
p.show()
```

Here is the plot (generated with MetaPost rather than Python, but from the same data):



where the $n = 0$ sample is marked with the prominent dot.

Pause to try 32. What would the plot look like for an exact solution of the continuous-time differential equations?

When the solution generates a true sine wave, which the continuous-time equations do, then the plot is of $v(t) = -\sin t$ versus $x(t) = \cos t$, which is a circle.

Since the discrete-time response is a growing spiral, it does not accurately represent the continuous-time system. Indeed after 50 or so time steps, the points have spiraled outward significantly. The spiraling signifies that $x[n]$ and $v[n]$ individually are not only oscillating, they are also growing.

6.1.2 Analysis using poles

To explain why the system oscillates and grows, find its poles. First, turn the two first-order equations into one second-order equation. Then find the poles of the second-order system. This method avoids having to understand what poles mean in a coupled system of equations.

To convert to a second-order system, use the first equation to eliminate v from the second equation. First multiply the second equation by T to get

$$Tv[n+1] - Tv[n] = -T^2x[n].$$

Now express $Tv[n+1]$ using $x[n]$ and $v[n]$; and $Tv[n]$ using $x[n-1]$ and $v[n-1]$. The forward-Euler replacements are

$$\begin{aligned} Tv[n+1] &= x[n+2] - x[n+1], \\ Tv[n] &= x[n+1] - x[n]. \end{aligned}$$

Making these replacements in $Tv[n+1] - Tv[n] = -T^2x[n]$ gives

$$\underbrace{(x[n+2] - x[n+1])}_{Tv[n+1]} - \underbrace{(x[n+1] - x[n])}_{Tv[n]} = -T^2x[n].$$

Collect like terms to get:

$$x[n+2] - 2x[n+1] + (1+T^2)x[n] = 0.$$

To get a system functional, we should have included an input signal or forcing function F. Physically, the forcing function represents the force driving the spring. Here is one way to add it:

$$x[n+2] - 2x[n+1] + (1+T^2)x[n] = f[n+2],$$

The system functional is:

$$\frac{X}{F} = \frac{1}{1 - 2\mathcal{R} + (1+T^2)\mathcal{R}^2}.$$

To find the poles, factor the denominator $1 - 2\mathcal{R} + (1+T^2)\mathcal{R}^2$ into the form $(1-p_1\mathcal{R})(1-p_2\mathcal{R})$.

Pause to try 33. Where are poles p_1 and p_2?

The quadratic formula is useful in finding p_1 and p_2, but too often it substitutes for, and does not augment understanding. So here is an alternative, intuitive analysis to find the poles. First expand the generic factored form:

$$(1 - p_1\mathcal{R})(1 - p_2\mathcal{R}) = 1 - (p_1 + p_2)\mathcal{R} + p_1p_2\mathcal{R}^2.$$

Now match this form to the particular denominator $1 - 2\mathcal{R} + (1 + T^2)\mathcal{R}^2$. The result is

$$\begin{aligned} p_1 + p_2 &= 2, \\ p_1p_2 &= 1 + T^2. \end{aligned}$$

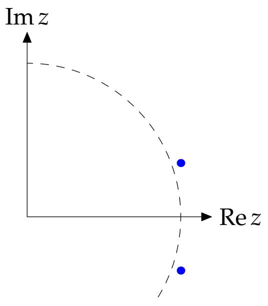
The sum of the roots is 2 while the product is greater than 1.

Pause to try 34. Show that these conditions are impossible to meet if p_1 and p_2 are real.

Let $p_1 = 1 + a$ and $p_2 = 1 - a$, which ensures that $p_1 + p_2 = 2$. Then $p_1p_2 = 1 - a^2$, which cannot be greater than 1 if a is real. So a must be imaginary. The resulting poles are:

$$p_{1,2} = 1 \pm jT$$

and are marked on the z-plane.



The poles are not on the positive real axis, which means that they produce oscillating outputs. Oscillation is desirable in a simulation of an oscillating continuous-time system. However, both poles lie outside the unit circle! Poles in that region of the z-plane produce growing outputs. So the poles of our system, which lie off the positive real axis and outside the unit circle, produce outputs that oscillate and grow, as shown in the X–V plot.

The forward-Euler method does not produce an accurate approximation to the continuous-time oscillating system.

Exercise 43. To place the poles on the unit circle, why not simulate with $T = 0$?

Exercise 44. On the z-plane, sketch how the poles move as T increases from 0 to 1.

6.2 Backward Euler

Let's find another method. Being lazy, we invent it using **symmetry**. If forward Euler is inaccurate, try backward Euler by estimating the derivatives using backward differences:

$$\begin{aligned} \dot{x}(nT) &\to \frac{x[n] - x[n-1]}{T}, \\ \dot{v}(nT) &\to \frac{v[n] - v[n-1]}{T}. \end{aligned}$$

These estimates are left-shifted versions of the forward-Euler estimates.

Then the system of continuous-time equations becomes

$$\begin{aligned} x[n] - x[n-1] &= Tv[n], \\ v[n] - v[n-1] &= -Tx[n]. \end{aligned}$$

The new values $x[n]$ and $v[n]$ depend on the new values themselves! This discrete-time system is an implicit recipe for computing the next samples, wherefore the backward Euler method is often called implicit Euler.

6.2.1 Finding an explicit recipe

Being a set of implicit equations, they require massaging to become an explicit recipe that we can program. You can do so with a matrix inversion, but let's do it step by step. The system of equations is

$$\begin{array}{rcrcl} x[n] & - & Tv[n] & = & x[n-1], \\ Tx[n] & + & v[n] & = & v[n-1]. \end{array}$$

Eliminate $v[n]$ to get an equation for $x[n]$ in terms of only the preceding samples $x[n-1]$ and $v[n-1]$. To eliminate $v[n]$, multiply the second equation by T then add. The result is

$$(1+T^2)x[n] = x[n-1] + Tv[n-1].$$

Similarly, eliminate $x[n]$ to get an equation for $v[n]$ in terms of only the preceding values $v[n-1]$ and $x[n-1]$. To eliminate $x[n]$, multiply the first equation by T then subtract. The result is

$$(1+T^2)v[n] = v[n-1] - Tx[n-1].$$

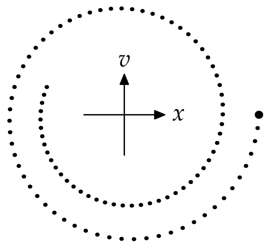
These equations are similar to the forward-Euler equations except for the factors of $1+T^2$. Those factors shrink $x[n]$ and $v[n]$, so they might control the runaway oscillations.

Pause to try 35. Modify the Python program for forward Euler to implement backward Euler, and plot the results.

To implement backward Euler, only two lines of the program need to be changed, the lines that compute the new samples. The code is

```
for n in range(N-1):
    x[n+1] = (x[n] + T*v[n])/(1+T**2)
    v[n+1] = (v[n] - T*x[n])/(1+T**2)
```

and the X–V plot is



Now the points spiral inward! The factor of $1+T^2$ is overkill.

6.2.2 Poles of backward Euler

Let's explain the inward spiral by finding the poles of the system. The first step is to convert the two first-order difference equations into one second-order equation.

Pause to try 36. Find the second-order difference equation.

To convert to a second-order difference equation, eliminate $v[n]$ and $v[n-1]$ by using

$$Tv[n] = x[n] - x[n-1]$$

and by using its counterpart shifted one sample, which is $Tv[n-1] = x[n-1] - x[n-2]$. Make these substitutions into $Tx[n] + v[n] = v[n-1]$ after multiplying both sides by $-T$. Then

$$\begin{aligned} -T^2x[n] &= \underbrace{(x[n]-x[n-1])}_{Tv[n]} - \underbrace{(x[n-1]-x[n-2])}_{Tv[n-1]} \\ &= x[n] - 2x[n-1] + x[n-2]. \end{aligned}$$

Combining the $x[n]$ terms and adding a forcing function F produces this difference equation

$$(1+T^2)x[n] - 2x[n-1] + x[n-2] = f[n]$$

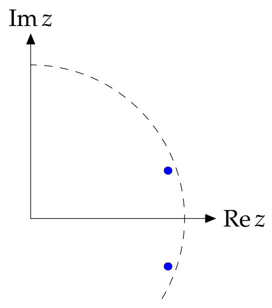
and this system functional

$$\frac{F}{X} = \frac{1}{(1+T^2) - 2\mathcal{R} + \mathcal{R}^2}.$$

Pause to try 37. Find the poles.

Now find its poles by factoring the denominator. Avoid the quadratic formula! The denominator looks similar to the denominator in forward Euler where it was $1 - 2\mathcal{R} + (1+T^2)\mathcal{R}^2$, but the end coefficients 1 and $1+T^2$ are interchanged. This interchange turns roots into their reciprocals, so the poles are

$$p_1 = \frac{1}{1+jT} \qquad p_2 = \frac{1}{1-jT}.$$



These poles lie inside the unit circle, so the oscillations die out and the X–V plot spirals into the origin.

Pause to try 38. Do a cheap hack to the program make the points stay on the unit circle. Hint: Add just eight characters to the code.

A cheap hack is to fix the problem manually. If dividing $x[n+1]$ and $v[n+1]$ by $1+T^2$ overcorrected, and dividing by 1 undercorrected, then try dividing by a compromise value $\sqrt{1+T^2}$:

```
for n in range(N-1):
    x[n+1] = (x[n] + T*v[n])/sqrt(1+T**2)
    v[n+1] = (v[n] - T*x[n])/sqrt(1+T**2)
```

However, this hack does not generalize, which is why it is a cheap hack rather than a method. In this problem we can solve the continuous-time system, so we can construct a hack to reproduce its behavior with a discrete-time system. However, for many systems we do not know the continuous-time solution, which is why we simulate. So we would like a principled method to get accurate simulations.

6.3 Leapfrog

Leapfrog, also known as the trapezoidal approximation, is a mixture of forward and backward Euler. Use forward Euler for the x derivative:

$$\dot{x}(nT) \to \frac{x[n+1]-x[n]}{T}$$

The discrete-time equation is, as in forward Euler,

$$x[n+1]-x[n]=Tv[n].$$

Then use backward Euler for the v derivative. So

$$\dot{v}(nT) \to \frac{v[n]-v[n-1]}{T}$$

and

$$v[n]-v[n-1]=-Tx[n]$$

or

$$v[n+1]-v[n]=-Tx[n+1].$$

In this mixed method, the x computation is an explicit recipe, whereas the v computation is an implicit recipe.

6.3.1 Simulation

Fortunately, this implicit recipe, unlike the full backward Euler, has a clean implementation. The system of equations is

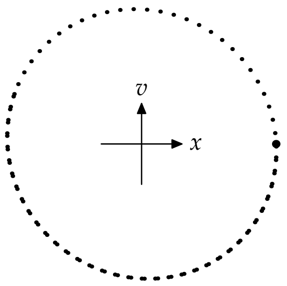
$$\begin{aligned} x[n+1] &= x[n]+Tv[n], \\ v[n+1] &= v[n]-Tx[n+1]. \end{aligned}$$

Pause to try 39. Implement leapfrog by modifying the magic two lines in the Python program.

The only change from forward Euler is in the computation of $v[n+1]$. Leapfrog uses $x[n+1]$, which is the just-computed value of x. So the code is

```
for n in range(N-1):
    x[n+1] = x[n] + T*v[n]
    v[n+1] = v[n] - T*x[n+1]
```

and the plot is



A beautiful circle without $\sqrt{1+T^2}$ hacks!

6.3.2 Analysis using poles

Let's explain that behavior by finding the poles of this system. As usual, convert the two first-order equations into one second-order equation for the position. To eliminate v, use the first equation, that $Tv[n] = x[n+1] - x[n]$. Then $v[n+1] = v[n] - Tx[n+1]$ becomes after multiplying by T:

$$\underbrace{(x[n+2]-x[n+1])}_{Tv[n+1]} - \underbrace{(x[n+1]-x[n])}_{Tv[n]} = -T^2x[n+1].$$

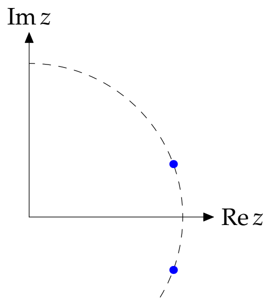
After rearranging and including a forcing function, the result is

$$x[n+2] - (2-T^2)x[n+1] + x[n] = f[n+2].$$

The system functional is

$$\frac{1}{1-(2-T^2)\mathcal{R}+\mathcal{R}^2}.$$

Again factor into the form $(1-p_1\mathcal{R})(1-p_2\mathcal{R})$. The product p_1p_2 is 1 because it is the coefficient of $\mathcal{R}^2$. The sum p_1+p_2 is $2-T^2$, which is less than 2. So the roots must be complex. A pair of complex-conjugate roots whose product is 1 lie on the unit circle. Poles on the unit circle produce oscillations that do not grow or shrink, wherefore leapfrog produces such a fine sine wave.



Exercise 45. Find the poles of the second-order equation and confirm that they lie on the unit circle.

6.4 Summary

The forward-Euler method is too aggressive. The backward-Euler method is too passive. But, at least for second-order systems, the mixed, forward–backward Euler method (leapfrog) is just right [14].

Recitation 7
Control

The goals of this chapter are to study:

- how to use feedback to control a system;
- how slow sensors destabilize a feedback system; and
- how to model inertia and how it destabilizes a feedback system.

A common engineering design problem is to control a system that integrates. For example, position a rod attached to a motor that turns input (control) voltage into angular velocity. The goal is an angle whereas the control variable, angular velocity, is one derivative different from angle. We first make a discrete-time model of such a system and try to control it without feedback. To solve the problems of the feedforward setup, we then introduce feedback and analyze its effects.

7.1 Motor model with feedforward control

We would like to design a controller that tells the motor how to place the arm at a given position. The simplest controller is entirely feedforward in that it does not use information about the actual angle. Then the high-level block diagram of the controller–motor system is

$$\begin{tikzcd}
\text{input} \arrow[r] & \boxed{\text{controller}} \arrow[r] & \boxed{\text{motor}} \arrow[r] & \text{output}
\end{tikzcd}$$

where we have to figure out what the output and input signals represent.

A useful input signal is the desired angle of the arm. This angle may vary with time, as it would for a robot arm being directed toward a teacup (for a robot that enjoys teatime).

The output signal should be the variable that interests us: the position (angle) of the arm. That choice helps later when we analyze feedback controllers, which use the output signal to decide what to tell the motor. With the output signal being the same kind of quantity as the input signal (both are angles), a feedback controller can easily compute the error signal by subtracting the output from the input.

With this setup, the controller–motor system takes the desired angle as its input signal and produces the actual angle of the arm as its output.

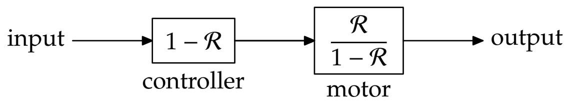
To design the controller, we need to model the motor. The motor turns a voltage into the arm's angular velocity ω. The continuous-time system that turns ω into angle is $\theta \propto \int \omega\, dt$. Its forward-Euler approximation is the difference equation

$$y[n] = y[n-1] + x[n-1].$$

The corresponding system functional is $\mathcal{R}/(1-\mathcal{R})$, which represents an accumulator with a delay.

Exercise 46. Draw the corresponding block diagram.

The ideal output signal would be a copy of the input signal, and the corresponding system functional would be 1. Since the motor's system functional is $\mathcal{R}/(1-\mathcal{R})$, the controller's should be $(1-\mathcal{R})/\mathcal{R}$. Sadly, time travel is not (yet?) available, so a bare $\mathcal{R}$ in a denominator, which represents a negative delay, is impossible. A realizable controller is $1-\mathcal{R}$, which produces a single delay $\mathcal{R}$ for the combined system functional:



Alas, the $1-\mathcal{R}$ controller is sensitive to the particulars of the motor and of our model of it. Suppose that the arm starts with a non-zero angle before the motor turns on (for example, the whole system gets rotated without the motor knowing about it). Then the output angle remains incorrect by this initial angle. This situation is dangerous if the arm belongs to a 1500-kg robot where an error of 10° means that its arm crashes through a brick wall rather than stopping to pick up the teacup near the wall.

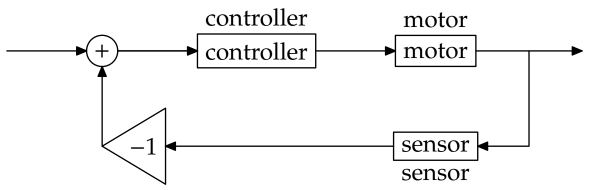
A problem in the same category is an error in the constant of proportionality. Suppose that the motor model underestimates the conversion between voltage and angular velocity, say by a factor of 1.5. Then the system functional of the controller–motor system is $1.5\mathcal{R}$ rather than $\mathcal{R}$. A 500-kg arm might again arrive at the far side of a brick wall.

One remedy for these problems is feedback control, whose analysis is the subject of the next sections.

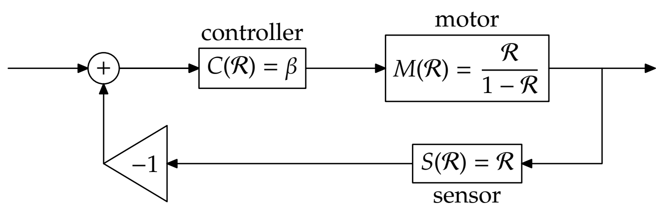
7.2 Simple feedback control

In feedback control, the controller uses the output signal to decide what to tell the motor. Knowing the input and output signals, an infinitely intelligent controller could deduce how the motor works. Such a controller would realize that the arm's angle starts with an offset or that the motor's conversion is incorrect by a factor of 1.5, and it would compensate for those and other problems. That mythical controller is beyond the scope of this course (and maybe of all courses). In this course, we use only linear-systems theory rather than strong AI. But the essential and transferable idea in the mythical controller is feedback.

So, sense the the angle of the arm, compare it to the desired angle, and use the difference (the error signal) to decide the motor's speed:



A real sensor cannot respond instantaneously, so assume the next-best situation, that the sensor includes one unit of delay. Then the sensor's output gets subtracted from the desired angle to get the error signal, which is used by the controller. The simplest controller, which uses so-called proportional control, just multiplies the error signal by a constant β. This setup has the block diagram


controller
$C(\mathcal{R}) = \beta$
motor
$M(\mathcal{R}) = \frac{\mathcal{R}}{1-\mathcal{R}}$
$S(\mathcal{R}) = \mathcal{R}$
sensor
-1
+

It was analyzed in lecture and has the system functional

$$\frac{C(\mathcal{R})M(\mathcal{R})}{1 + C(\mathcal{R})M(\mathcal{R})S(\mathcal{R})} = \frac{\beta\mathcal{R}/(1-\mathcal{R})}{1+\beta\mathcal{R}^2/(1-\mathcal{R}).}$$

Multiply by $(1-\mathcal{R})/(1-\mathcal{R})$ to clear the fractions. Then

$$F(\mathcal{R}) = \frac{\beta\mathcal{R}}{1-\mathcal{R}+\beta\mathcal{R}^2},$$

where $F(\mathcal{R})$ is the functional for the whole feedback system.

Let's analyze its behavior in the extreme cases of the gain β. As $\beta \to \infty$, the system functional limits to $\mathcal{R}/\mathcal{R}^2 = 1/\mathcal{R}$, which is a time machine. Since we cannot build a time machine just by choosing a huge gain in a feedback system, some effect should prevent us raising $\beta \to \infty$. Indeed, instability will prevent it, as we will see by smoothly raising β from 0 to ∞.

To study stability, look at the poles of the feedback system, which are given by the factors of the denominator $1-\mathcal{R}+\beta\mathcal{R}^2$. The factored form is $(1-p_1\mathcal{R})(1-p_2\mathcal{R})$. So the sum of the poles is 1 and their product is β. At the $\beta \to 0$ extreme, which means no feedback, the poles are approximately $1-\beta$ and β. The pole near 1 means that the system is almost an accumulator; it approximately integrates the input signal. This behavior is what the motor does without feedback and is far from our goal that the controller–motor system copy the input to the output.

Turning up the gain improves the control. However, at the $\beta \to \infty$ extreme, the poles are roughly at

$$p_{1,2} \approx \frac{1}{2} \pm j\sqrt{\beta}$$

so that $p_1p_2 = \beta$. Those poles lie far outside the unit circle, making the system highly unstable. *Too much gain destabilizes a feedback system.*

To find the best gain, study the poles. The pole farthest from the origin has the most rapidly growing, or most slowly decaying output. Therefore, to make the system as stable as possible, minimize the distance to the pole most distant from the origin. To do so, place poles at the same location. In this example, the location must be $p_1 = p_2 = 1/2$ since $p_1 + p_2 = 1$. Since $\beta = p_1p_2$, choose $\beta = 1/4$. Now the output position rapidly approaches the desired position. Equivalently, in response to an impulse input signal, the error signal decays rapidly to zero, roughly halving each time step.

Exercise 47. Why does the error signal *roughly* halve, rather than exactly halve with every time step?

7.3 Sensor delays

The preceding model contained a rapid sensor. Suppose instead that the sensor is slow, say $S(R) = \mathcal{R}^2$.

Pause to try 40. With this sensor, what is the functional for the feedback system?

The functional for the feedback system is

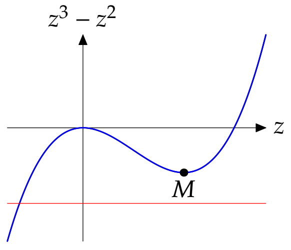
$$\frac{\beta\mathcal{R}}{1-\mathcal{R}+\beta\mathcal{R}^3},$$

which is the previous functional with the $\mathcal{R}^2$ in the denominator replaced by $\mathcal{R}^3$ because of the extra power of $\mathcal{R}$ in the sensor functional. There are many analyses that one can do on this system. For simplicity, we choose a particular gain β – the rapid-convergence gain with the fast sensor – and see how the extra sensor delay moves the poles. But before analyzing, predict the conclusion!

Pause to try 41. Will the extra sensor delay move the least stable pole inward, outward, or leave its magnitude unchanged?

The poles are at the roots of the corresponding equation $z^3 - z^2 + \beta = 0$ or

$$z^3 - z^2 = -\beta.$$



Here is a sketch of the curve $z^3 - z^2$. Its minimum is at $M = (2/3, -4/27)$, so the horizontal line at $-1/4$ intersects the curve only once, in the left half of the plane. The equation therefore has one (negative) real root and two complex roots. So, for $\beta = 1/4$, the system has two complex poles and one real pole. The following Python code finds the poles. It first finds the real pole p_1 using the Newton–Raphson [15] method of successive approximation. The Newton–Raphson code is available as part of the `scipy` package. The real pole constrains the real part of the complex poles because the sum of the poles $p_1 + p_2 + p_3$ is 1, and the two complex poles have the same real part. So

$$\operatorname{Re} p_{2,3} = \frac{1-p_1}{2}.$$

To find the imaginary part of p_2 or p_3, use the product of the poles. The product $p_1p_2p_3$ is $-\beta$. Since the magnitudes of the complex poles are equal because they are complex conjugates, we have

$$|p_{2,3}| = \sqrt{\frac{-\beta}{p_1}}.$$

Then find the imaginary part of one complex pole from the computed real part and magnitude. This algorithm is implemented below.

```
from scipy import *
```

```
def poly(beta):
    def f(z):  # closure that knows the passed-in value of beta
        return z**3-z**2+beta
    return f

# return the three poles of 1-R+beta*R^3 by solving z^3-z^2+beta=0
# method works for beta>4/27 (one real root, two complex roots)
def solve(beta):
    # use Newton-Raphson to find the one real root (for beta>4/27)
    realroot = optimize.newton(poly(beta), 1.0)
    # use realroot to find complex roots
    realpart = (1-realroot)/2           # sum of the roots is 1
    magnitude = sqrt(-beta/realroot)   # product of roots is -beta
    imaginarypart = sqrt(magnitude**2 - realpart**2)
    complexroot = realpart + 1j*imaginarypart
    return (realroot, complexroot, conjugate(complexroot))
```

The result is

$$p_1 \approx -0.419,$$
$$p_2 \approx 0.710 + 0.303j,$$
$$p_3 \approx 0.710 - 0.303j.$$

With these locations, the complex poles are the least stable modes. These poles have a magnitude of approximately 0.772. In the previous system with the fast sensor and the same gain $\beta = 1/4$, both poles had magnitude 0.5. So the sensor delay has made the system more unstable and, since the poles are complex, has introduced oscillations.

To make the system more stable, one can reduce β. But this method has problems. The $\beta \to \infty$ limit makes the feedback system turn into the system $\mathcal{R}^{-2}$, independent of the motor's characteristics. The other direction, reducing β, exposes more particulars of the motor, making a feedback system sensitive to the parameters of the motor. Thus lower β means one gives up some advantages of feedback. No choices are easy if the sensor delay is long. When β is small, the system system is stable but benefits hardly at all from feedback.

Pause to try 42. What happens when β is large?

When β is large, then the feedback system is less stable and eventually unstable. To prove this, look at how the denominator $1 - \mathcal{R} + \beta\mathcal{R}^3$ constrains the location of the poles. The product of the poles the negative of the coefficient of $\mathcal{R}^3$, so $p_1p_2p_3 = -\beta$. Using magnitudes,

$$|p_1|\,|p_2|\,|p_3| = \beta,$$

so when $\beta > 1$, at least one pole must have magnitude greater than one, meaning that it lies outside the unit circle.

From the analysis with $S(R) = \mathcal{R}$ and $S(R) = \mathcal{R}^2$, try to guess what happens with one more delay in the sensor, which makes $S(R) = \mathcal{R}^3$ (again with $\beta = 1/4$).

Exercise 48. What happens to the stability if the sensor has yet another delay, so $S(\mathcal{R}) = R^3$? First guess then check your guess by finding the poles numerically or otherwise.

7.4 Inertia

Now return to the quick sensor with one delay, but improve the physical model of the motor. A physical motor cannot change its angular velocity arbitrarily quickly, especially with a heavy rod attached to it. Instant changes imply zero inertia. So we should add inertia to the model of the motor.

A simple model of inertia is a new term in the difference equation:

$$y[n] = y[n-1] + x[n-1] + \underbrace{\frac{1}{2}(y[n-1] - y[n-2])}_{\text{inertia}}.$$

The difference $y[n-1] - y[n-2]$ estimates the motor's angular velocity. The coefficient of 1/2 means that, with each time step, the motor gets rid of one-half of its previous angular velocity. Alternatively, it means that one-half of its previous angular velocity remains to affect the new angle. This coefficient depends on the mass of the motor and the mass and length of the rod – more exactly, on the moment of inertia of the system – and on the power of the motor. For illustrating the ideas, we picked the convenient coefficient of 1/2.

The motor's system functional, which was $\mathcal{R}/(1-\mathcal{R})$, becomes

$$M(\mathcal{R}) = \frac{\mathcal{R}}{1 - \frac{3}{2}\mathcal{R} + \frac{1}{2}\mathcal{R}^2}.$$

Pause to try 43. Find the poles of this system and mark them on a pole–zero diagram.

This functional factors into

$$M(\mathcal{R}) = \frac{\mathcal{R}}{(1 - \frac{1}{2}\mathcal{R})(1-\mathcal{R})}$$

so the poles are $p_1 = 1/2$ and $p_2 = 1$.

The functional for the feedback system of controller $C(\mathcal{R})$, sensor $S(\mathcal{R})$, and motor $M(\mathcal{R})$ is

$$F(\mathcal{R}) = \frac{C(\mathcal{R})M(\mathcal{R})}{1 + C(\mathcal{R})M(\mathcal{R})S(\mathcal{R})},$$

With the usual controller $C(\mathcal{R}) = \beta$, fast sensor $S(\mathcal{R}) = \mathcal{R}$, and new motor model $M(\mathcal{R})$ with inertia, the feedback system has the functional

$$\frac{\beta\mathcal{R}/(1 - \frac{3}{2}\mathcal{R} + \frac{1}{2}\mathcal{R}^2)}{1 + \beta\mathcal{R}^2/(1 - \frac{3}{2}\mathcal{R} + \frac{1}{2}\mathcal{R}^2)}.$$

Clear the fractions to get

$$F(\mathcal{R}) = \frac{\beta\mathcal{R}}{1 - \frac{3}{2}\mathcal{R} + \left(\beta + \frac{1}{2}\right)\mathcal{R}^2}.$$

This denominator is quadratic so we can find the poles for all β without needing numerical solutions. So let β increase from 0 to ∞. Their locations are determined by factoring the denominator When $\beta = 0$, it factors into $(1 - \mathcal{R}/2)(1 - \mathcal{R})$, and the poles are at 1/2 and 1 – which are the poles of the motor itself. The pole at 1 indicates an accumulator, which means that the system is very different than one that copies the input signal to the output signal. But we knew it would happen that way, because choosing $\beta = 0$ turns off feedback.

As β increases, the poles move. The sum $p_1 + p_2$ remains constant at 3/2, so the poles are at $3/4 \pm \alpha$. For $\beta = 0$, the α is 1/4. As β increases, α increases and the poles slide along the real axis until they collide at $p_{1,2} = 3/4$. When they collide, the product of the poles is $p_1 p_2 = 9/16$. This product is the coefficient of $\mathcal{R}^2$, which is $1/2 + \beta$. So $1/2 + \beta = 9/16$, which means that the poles collide when $\beta = 1/16$. That controller gain results in the most stable system. It is also significantly smaller than the corresponding gain when the motor had no inertia. This simple controller that only has a gain has difficulty compensating for inertia.

Pause to try 44. For what β do the poles cross the unit circle into instability? Compare that critical β with the corresponding value in the model without inertia.

As β increases farther, the poles move along a vertical line with real part 3/4. The next interesting β is when the poles hit the unit circle. Their product is then 1, which is the coefficient of $\mathcal{R}^2$ in the denominator of the system functional. So $1/2 + \beta = 1$ or $\beta = 1/2$. The resulting poles are

$$p_{1,2} = \frac{3}{4} \pm j\frac{\sqrt{7}}{4}.$$

In the model without inertia, β could increase to 1 before the feedback system became unstable, whereas now it can increase only till 1/2: *Inertia destabilizes the feedback system.*

Exercise 49. Sketch how the poles move as β changes from 0 to ∞.

Exercise 50. What if the system has more inertia, meaning that old angular velocities persist longer? For example:

$$y[n] = y[n-1] + x[n-1] + \underbrace{\frac{4}{5}(y[n-1] - y[n-2])}_{\text{inertia}}.$$

Sketch how the poles of the feedback system move as β changes from 0 to ∞, and compare with the case of no inertia and of inertia with a coefficient of 1/2.

Recitation 8
Proportional and derivative control

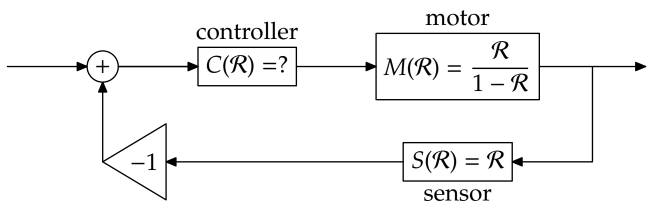
The goals of this chapter are:

- to introduce derivative control; and
- to study the combination of proportional and derivative control for taming systems with integration or inertia.

The controllers in the previous chapter had the same form: The control signal was a multiple of the error signal. This method cannot easily control an integrating system, such as the motor positioning a rod even without inertia. If the system has inertia, the limits of proportional control become even more apparent. This chapter introduces an alternative: derivative control.

8.1 Why derivative control

An alternative to proportional control is derivative control. It is motivated by the integration inherent in the motor system. We would like the feedback system to make the actual position be the desired position. In other words, it should copy the input signal to the output signal. We would even settle for a bit of delay on top of the copying. This arrangement is shown in the following block diagram:



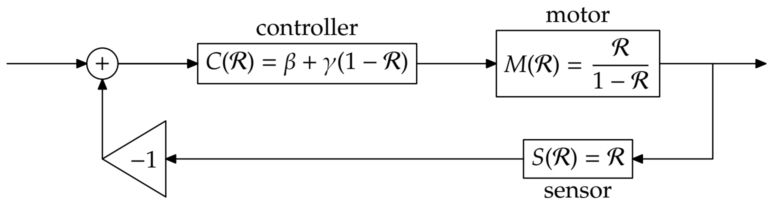
controller
$C(\mathcal{R}) = ?$
motor
$M(\mathcal{R}) = \dfrac{\mathcal{R}}{1-\mathcal{R}}$
$+$
-1
$S(\mathcal{R}) = \mathcal{R}$
sensor

Since the motor has the functional $\mathcal{R}/(1-\mathcal{R})$, let's put a discrete-time derivative $1-\mathcal{R}$ into the controller to remove the $1-\mathcal{R}$ in the motor's denominator. With this **derivative control**, the forward-path cascade of the controller and motor contains only powers of $\mathcal{R}$. Although this method is too fragile to use alone, it is a useful idea. Pure derivative control is fragile because it uses pole–zero cancellation. This cancellation is mathematically plausible but, for the reasons explained in lecture, it produces unwanted offsets in the output. However, derivative control is still useful. As we will find, in combination with proportional control, it helps to stabilize integrating systems.

8.2 Mixing the two methods of control

Proportional control uses β as the controller. Derivative control uses $\gamma(1-\mathcal{R})$ as the controller. The linear mixture of the two methods is

$$C(\mathcal{R}) = \beta + \gamma(1-\mathcal{R}).$$


controller
$C(\mathcal{R}) = \beta + \gamma(1-\mathcal{R})$
motor
$M(\mathcal{R}) = \frac{\mathcal{R}}{1-\mathcal{R}}$
+
−1
$S(\mathcal{R}) = \mathcal{R}$
sensor

Let $F(\mathcal{R})$ be the functional for the entire feedback system. Its numerator is the forward path $C(\mathcal{R})M(\mathcal{R})$. Its denominator is $1 - L(\mathcal{R})$, where $L(\mathcal{R})$ is the loop functional or **loop gain** that results from going once around the feedback loop. Here the loop functional is

$$L(\mathcal{R}) = -C(\mathcal{R})M(\mathcal{R})S(\mathcal{R}).$$

Don't forget the contribution of the inverting (gain= −1) element! So the overall system functional is

$$F(\mathcal{R}) = \frac{(\beta + \gamma(1-\mathcal{R}))\,\frac{\mathcal{R}}{1-\mathcal{R}}}{1 + (\beta + \gamma(1-\mathcal{R}))\,\frac{\mathcal{R}}{1-\mathcal{R}}\mathcal{R}}.$$

Clear the fractions to get

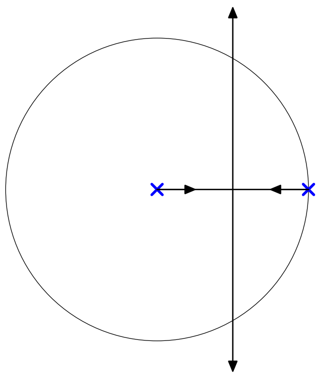
$$F(\mathcal{R}) = \frac{\text{whatever}}{1 - \mathcal{R} + (\beta + \gamma(1-\mathcal{R}))\mathcal{R}^2}.$$

The *whatever* indicates that we don't care what is in the numerator. It can contribute only zeros, whereas what we worry about are the poles. The poles arise from the denominator, so to avoid doing irrelevant algebra and to avoid cluttering up the expressions, we do not even compute the numerator as long as we know that the fractions are cleared.

The denominator is

$$1 - \mathcal{R} + (\beta + \gamma)\mathcal{R}^2 - \gamma\mathcal{R}^3.$$

This cubic polynomial produces three poles. Before studying their locations – a daunting task with a cubic – do an extreme-cases check: Take the limit $\gamma \to 0$ to turn off derivative control. The system should turn into the pure proportional-control system from the previous chapter. It does: The denominator becomes $1 - \mathcal{R} + \beta\mathcal{R}^2$, which is the denominator from Section 7.2. As the proportional gain β increases from 0 to ∞, the poles, which begin at 0 and 1, move inward; collide at 1/2 when $\beta = 1/4$; then split upward and downward to infinity. Here is the root locus of this limiting case of $\gamma \to 0$, with only proportional control:



8.3 Optimizing the combination

We would like to make the whole system as stable as possible, in the sense that the least stable pole is as close to the origin as possible. The root locus for the general combination has three branches, one for each pole, whereas the limiting case of proportional control has only two poles and two branches. Worse, the root locus for the general combination is generated by two parameters – the gains of the proportional and the derivative portions – whereas in the limiting case it is generated by only one parameter. The general analysis seems difficult.

Surprisingly, the extra parameter rescues us from painful mathematics. To see how, look at the coefficients in the cubic:

$$1 - \mathcal{R} + (\beta + \gamma)\mathcal{R}^2 - \gamma\mathcal{R}^3.$$

The factored form is

$$(1 - p_1\mathcal{R})(1 - p_2\mathcal{R})(1 - p_3\mathcal{R}) = 1 - \underbrace{(p_1 + p_2 + p_3)}_{1}\mathcal{R} + \underbrace{(p_1p_2 + p_1p_3 + p_2p_3)}_{\beta+\gamma}\mathcal{R}^2 - \underbrace{p_1p_2p_3}_{\gamma}\mathcal{R}^3.$$

So the first constraint is

$$p_1 + p_2 + p_3 = 1,$$

showing that the center of gravity of the poles is 1/3. That condition is independent of β and γ. So the most stable system has a triple pole at 1/3, if that arrangement is possible. To see why that arrangement is the most stable, imagine starting from it. Now move one pole inward along the real axis to increase its stability. To preserve the invariant $p_1 + p_2 + p_3 = 1$, at least one of the other poles must move outward and become less stable. Thus it is best not to move any pole away from the triple cluster, so it is the most stable arrangement.

Exercise 51. Where does the preceding argument require that the center of gravity be independent of β and γ?

If the triple-pole arrangement is impossible, then the preceding argument, which assumed its existence, does not work. And we need lots of work to find the best arrangement of poles.

Fortunately, the triple pole is possible thanks to the extra parameter γ. Having freedom to choose β and γ, we can set the $\mathcal{R}^2$ coefficient $\beta+\gamma$ independently from the $\mathcal{R}^3$ coefficient, which is $-\gamma$. So, using β and γ as separate dials, we can make any cubic whose poles are centered on 1/3.

Let's set those dials by propagating constraints. With $p_1 = p_2 = p_3 = 1/3$, the product $p_1p_2p_3 = 1/27$. So the gain of the derivative controller is

$$\gamma = \frac{1}{27}.$$

The last constraint is that $p_1p_2 + p_1p_3 + p_2p_3 = 3/9 = 1/3$. So $\beta + \gamma = 1/3$. With $\gamma = 1/27$, this equation requires that the gain of the proportional controller be $\beta = 8/27$. The best controller is then

$$C(\mathcal{R}) = \frac{8}{27} + \frac{1}{27}(1-\mathcal{R}) = \frac{1}{3}\left(1-\frac{\mathcal{R}}{9}\right).$$

Exercise 52. What is the pole-zero plot of the forward path $C(\mathcal{R})M(\mathcal{R})$?

This controller has a zero at $z = 1/9$. So the added zero has pulled the poles into the sweet spot of 1/3. In comparison with pure proportional control, where the worst pole could not get closer than $z = 1/2$, derivative control has dragged the poles all the way to $z = 1/3$. A judicious amount of derivative control has helped stabilize the system.

8.4 Handling inertia

The last example showed how to use derivative control and computed how much to use. However, derivative control was not essential to stabilizing the feedback system since proportional control alone can do so and can drag the least stable pole to $z = 1/2$. But derivative control becomes essential when the system has inertia.

Without inertia, the motor accumulates angular velocity to produce angle, which is represented by the difference equation

$$y[n] = y[n-1] + x[n-1]$$

and the system functional $M(\mathcal{R}) = \mathcal{R}/(1-\mathcal{R})$. The model of inertia in Section 7.4 added a term to the motor's difference equation:

$$y[n] = y[n-1] + x[n-1] + \underbrace{m(y[n-1] - y[n-2])}_{\text{inertia}},$$

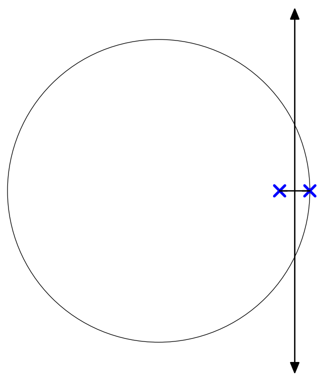
where m is a constant between 0 (no inertia) and 1 (maximum inertia). This term changes the motor's system functional to

$$M(\mathcal{R}) = \frac{1}{1-(1+m)\mathcal{R}+m\mathcal{R}^2}.$$

It factors into poles at m and 1:

$$M(\mathcal{R}) = \frac{1}{(1-m\mathcal{R})(1-\mathcal{R})}.$$

The analysis in **Section 7.4** used $m = 1/2$, and then asked you to try $m = 4/5$. You should have found that the arm is hard to position when m is so close to 1. The figure shows the root locus for the motor with inertia $m = 4/5$ and controlled only using proportional control. The least stable pole can, with the right proportional gain, be dragged to the collision point $z = 0.9$. But the pole cannot be moved farther inward without moving the other pole outward. A pole at $z = 0.9$ means that the system's response contains the mode 0.9^n, which converges only slowly to zero.



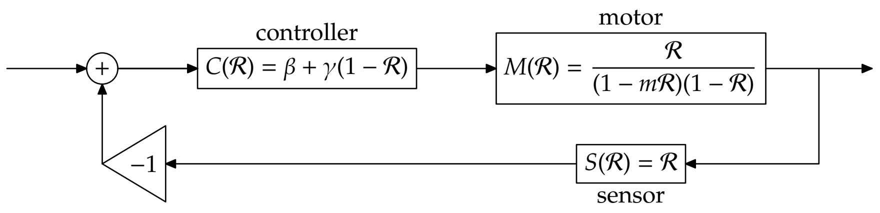
Pause to try 45. How many time steps before 0.9^n has decayed roughly by a factor of e^3 (commonly used as a measure of 'has fallen very close to zero')?

The decay 0.9^n takes roughly 10 steps to fall by a factor of e. Use the greatest approximation in mathematics:

$$0.9^{10} = (1 - 0.1)^{10} \approx e^{-0.1\times 10} = e^{-1}.$$

So 30 time steps make the signal fall by a factor of e^3. In some applications, this wait might be too long.

Derivative control can pull the poles toward the origin, thereby hastening the convergence. Let's analyze how much derivative control to use by finding the poles of the feedback system. The feedback system is


controller
$C(\mathcal{R}) = \beta + \gamma(1-\mathcal{R})$
motor
$M(\mathcal{R}) = \dfrac{\mathcal{R}}{(1-m\mathcal{R})(1-\mathcal{R})}$
-1
$S(\mathcal{R}) = \mathcal{R}$
sensor
+

Its system functional has the form

$$F(\mathcal{R}) = \frac{N(\mathcal{R})}{D(\mathcal{R})},$$

where the denominator is

$$\begin{aligned} D(\mathcal{R}) &= 1 - \underbrace{(-C(\mathcal{R})M(\mathcal{R})S(\mathcal{R}))}_{\text{loop functional } L(\mathcal{R})} \\ &= 1 + C(\mathcal{R})M(\mathcal{R})S(\mathcal{R}). \end{aligned}$$

In the product $C(\mathcal{R})M(\mathcal{R})S(\mathcal{R})$, the only term with a denominator is $M(\mathcal{R})$. To clear its denominator from $D(\mathcal{R})$, the whole denominator will get multiplied by the denominator of $M(\mathcal{R})$, which is $(1 - m\mathcal{R})(1 - \mathcal{R})$. So the system functional will end up with a denominator of

$$(1 - m\mathcal{R})(1 - \mathcal{R}) + \underbrace{(\beta + \gamma(1 - \mathcal{R}))}_{\text{controller}}\mathcal{R}^2.$$

After the controller come two powers of $\mathcal{R}$, one from the sensor, the other from the numerator of the motor functional $M(\mathcal{R})$. After expanding the products, the denominator is

$$1-(1+m)\mathcal{R}+(m+\beta+\gamma)\mathcal{R}^2-\gamma\mathcal{R}^3.$$

This system has three parameters: the proportional gain β, the derivative gain γ, and the inertia pole m. Before spending the effort to analyze a cubic equation for its poles, check whether the equation is even reasonable! The fastest check is the extreme cases of taking parameters to zero. The limit $m \to 0$ wipes out the inertia and should reproduce the denominator in the preceding section. In that limit, the denominator becomes

$$1-\mathcal{R}+(\beta+\gamma)\mathcal{R}^2-\gamma\mathcal{R}^3 \qquad (m \to 0 \text{ limit}),$$

which matches the denominator in **Section 8.2**. Good!

Adding the limit $\gamma \to 0$ then wipes out derivative control, which should reproduce the analysis of the simple motor with only proportional control in **Section 7.2**. Adding the $\gamma \to 0$ limit turns the denominator into

$$1-\mathcal{R}+\beta\mathcal{R}^2 \qquad (m \to 0, \gamma \to 0 \text{ limit}),$$

which passes the test. Adding the $\beta \to 0$ limit wipes out the remaining feedback, leaving the bare motor functional $M(\mathcal{R})$, which indeed has a factor of $1-\mathcal{R}$ in the denominator. So the candidate denominator passes this third test too.

Although passing three tests does not guarantee correctness, the tests increase our confidence in the algebra, perhaps enough to make it worthwhile to analyze the cubic to find where and how to place the poles. For convenience, here is the cubic again:

$$1-(1+m)\mathcal{R}+(m+\beta+\gamma)\mathcal{R}^2-\gamma\mathcal{R}^3.$$

We would like to choose β and γ so that the worst pole – the one farthest from the origin – is as close as possible to the origin.

Maybe we can try the same trick (method?) that we used in the analysis without inertia: to place all three poles at the same spot. Let's assume that this solution is possible, and propagate constraints again. The sum of the poles is $1+m$, so each pole is at $p=(1+m)/3$. The product of the poles, p^3, is $(1+m)^3/27$, which tells us

$$\gamma=\frac{(1+m)^3}{27}.$$

. The sum of pairwise products of poles is $3p^2$ and is therefore $m+\beta+\gamma$. Since $3p^2$ is $(1+m)^2/3$, the equation for β is

$$\frac{(1+m)^2}{3}=m+\beta+\gamma.$$

So the proportional gain is:

$$\beta=\frac{(1+m)^2}{3}-m-\gamma=\frac{m^2-m+1}{3}-\frac{(1+m)^3}{27}.$$

To summarize,

$$\gamma = \frac{(1+m)^3}{27},$$
$$\beta = \frac{m^2-m+1}{3} - \frac{(1+m)^3}{27}.$$

An interesting special case is maximum inertia, which is $m = 1$. Then $\gamma = 8/27$ and $\beta = 1/27$, so the controller is

$$\begin{aligned}\frac{1}{27} + \frac{8}{27}(1-\mathcal{R}) &= \frac{1}{3} - \frac{8}{27}\mathcal{R}\\ &= \frac{1}{3}\left(1 - \frac{8}{9}\mathcal{R}\right).\end{aligned}$$

So the controller contains a zero at 8/9, near the double pole at 1. This mixed proportional–derivative controller moves all the poles to $z = (1+m)/3 = 2/3$, which is decently inside the unit circle. So this mixed controller can stabilize even this hard case. This case is the hardest one to control because the motor-and-rod system now contains two integrations: one because the motor turns voltage into angular velocity rather than position, and the second because of the inertia pole at 1. This system has the same loop functional as the steering-a-car example in lecture (!), which was unstable for any amount of pure proportional gain. By mixing in derivative control, all the poles can be placed at 2/3, which means that the system is stable and settles reasonably quickly. Since

$$\left(\frac{2}{3}\right)^{2.5} \approx e^{-1},$$

the time constant for settling is about 2.5 time steps, and the system is well settled after three time constants, or about 7 time steps.

8.5 Summary

To control an integrating system, try derivative control. To control a system with inertia, also try derivative control. In either situation, do not use pure derivative control, for it is too fragile. Instead, mix proportional and derivative control to maximize the stability, which often means putting all the poles on top of each other.

Recitation 9

Image processing using operators

The goals of this chapter are:

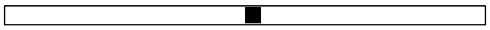
- to analyze spatial signals using functionals (or operators);
- to look in slow motion at how to invert blurring operations;
- to notice non-idealities, such as quantization error, that occur when dealing with messy signals from the world.

All our signals so far have been signals that depend on time, and time is a special quantity. First, time has a direction. This mysterious idea is incorporated into a proverb: 'It is difficult to make predictions, especially about the future.' Therefore, delay elements ($\mathcal{R}$), which use past data to future outputs, are natural objects to use when analyzing time signals. Systems built only out of $\mathcal{R}$ operators are causal systems: The output cannot become nonzero before the input becomes nonzero. In contrast, space has no preferred direction. If the right-shift operator $\mathcal{R}$ is useful for analyzing spatial signals, then the left-shift operator $\mathcal{L}$ should be equally useful. Systems built only out of $\mathcal{L}$ operators are anticausal systems. In general, systems that process spatial signals are composed out of $\mathcal{R}$ and $\mathcal{L}$ operators, and these systems are neither causal nor anticausal; they are non-causal. In this chapter we use non-causal systems to blur and unblur images.

A second mysterious feature of time is that the universe knows only one time axis, whereas it knows multiple space axes. Therefore, a time signal can be analyzed using $\mathcal{R}$, but a spatial signal might need right-shift and left-shift (by symmetry) but also up-shift and down-shift (for the second dimension) and back-shift and forward-shift operators (for the third dimension). Images are often two dimensional, so you can often dispense with back- and forward-shift operators. But the other four operators are essential for image processing. In order to focus on the main idea of bidirectionality, however, we limit this chapter to operations that treat each row independently, meaning that we need only right- and left-shift operators.

9.1 Causal blurring

Let's first blur a simple picture, the impulse:



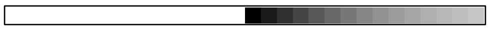
It is one image row with 30 samples (pixels). The color represents the value at that pixel, with black being 1 and white being 0. In most image formats, intensities are normally represented in reverse, with black being 0 and white being 1. However, that choice wastes tons of toner (think of 180 copies of many pages each with lots of black ink on it), so we use the less frequent but environmentally friendlier convention.

9.1.1 Blurred impulse

Now feed the impulse picture through a causal blurring system:

$$\frac{1}{1-0.9\mathcal{R}} = 1 + 0.9\mathcal{R} + 0.9^2\mathcal{R}^2 + \cdots.$$

The result is

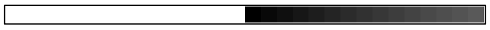


The impulse has become blurry.

Let's check this new picture by examining its features. First, the left side is still white (meaning 0), which is as it should be. The system contains only $\mathcal{R}$, so it can be rewritten as powers only of $\mathcal{R}$. Each $\mathcal{R}$ shifts the black pixel to the right, so only the right half of the processed image should have ink. Great!

A second feature is that pixel values become less black toward the right. Great! Each term in the polynomial expansion comes with as many powers of 0.9 as of $\mathcal{R}$. So as one moves right by n pixels, the intensities are multiplied by 0.9^n, making the right side transition toward white. However, even by the end of the row the color is not fully white. This color is also as it should be. The blurring system has a pole, which means that the impulse response lasts forever. So we would need an infinitely long row before the pixel value returned to white. That the finite-length row does not show this ideal behavior is an example of an edge effect.

For comparison, here is extreme blurring:

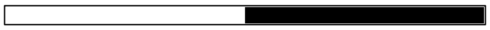


which shows the impulse response of the system

$$\frac{1}{1-0.97\mathcal{R}} = 1 + 0.97\mathcal{R} + 0.97^2\mathcal{R}^2 + \cdots.$$

9.1.2 Blurred step

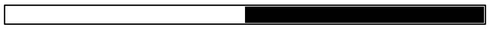
The impulse is the simplest picture. Now play with the next-simplest picture:



It is the step picture: 0 on the left and 1 on the right. After passing through the system

$$\frac{1}{1-0.9\mathcal{R}} = 1 + 0.9\mathcal{R} + 0.9^2\mathcal{R}^2 + \cdots$$

it becomes



This output looks identical to the input! Why did the system do nothing? To see the source of the problem, look at the output pixel values. The system implements a decaying weighted average with decay constant 0.9. So it takes roughly 10 steps for the weight to decay to $1/e$. This so-called length constant suggests that, roughly speaking, the system sums the last 10 input samples. Its input is the step function $1, 1, 1, 1, 1, \ldots$. So the output samples build toward 10, wherein lies the problem. The image representation can display pixel values only between 0 (white) and 1 (black). Any value above 1 is black again, and any negative value is just white.

To fix this problem of clipping, we need to squeeze the pixel values into 0 to 1. There are at least two methods. One method is to look at all the output pixel values, find the maximum value, and rescale every value so that this maximum value is 1. This method is the hack method.

A principled alternative is to investigate the output using the signals-and-systems tools. The input most likely to overflow the 0 to 1 output range is the step function. So, what is the output of the system when the step function is the input? One way to find the output is to imagine a cascade of an accumulator followed by the blurring system. When fed an impulse, the accumulator outputs a step function. So the impulse response of the cascade is also the response of the blurring system to the step function.

To find the impulse response of the cascade, write its system functional:

$$\frac{1}{1-\mathcal{R}}\,\frac{1}{1-0.9\mathcal{R}}.$$

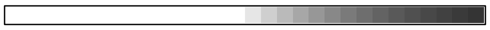
Now expand in partial fractions:

$$\frac{1}{1-\mathcal{R}}\,\frac{1}{1-0.9\mathcal{R}} = 10\left(\frac{1}{1-\mathcal{R}} - \frac{0.9}{1-0.9\mathcal{R}}\right).$$

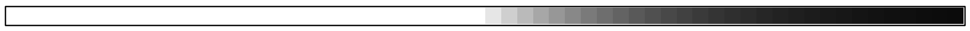
So the output samples are $10\cdot(1-0.9^{n+1})$ for $n \geq 0$ (and 0 otherwise). For $n = 0$, the output is 1. But as n grows, the 0.9^{n+1} term goes to zero, making the output 10: far outside the color range. So let's modify the system by dividing the output by 10. Then

$$\text{non-clipping blurring functional} = \frac{0.1}{1-0.9\mathcal{R}}.$$

When fed the step function, its output is

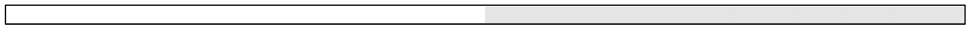


Now the output looks reasonable: The colors, after tens of pixels, approach the steady-state color, which is black. Even at the right edge, the pixels are far from black, so here is a longer row (60 pixels) to show a closer approach to black:

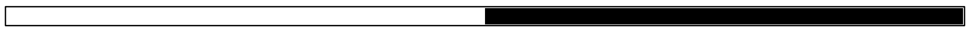


9.2 Causal unblurring

Let's undo the blurring. The blurring system uses a pole at 0.9. So use a zero at 0.9 to cancel this pole (after saying a silent prayer to the gods of pole–zero cancellation). Here is the result of feeding the blurred step function into a zero at 0.9:

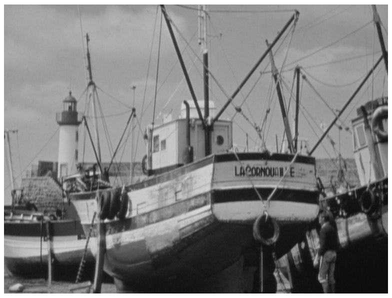


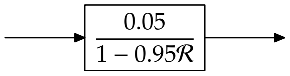
Whoops: The color is gray rather than black. The problem is again the gain. The blurring system needed, and used, a gain of 0.1 to keep the outputs within the range $[0, 1]$. So the unblurring system needs a gain of 10. Thus the unblurring system is $10(1-0.9\mathcal{R})$. When fed the blurry step, its output is

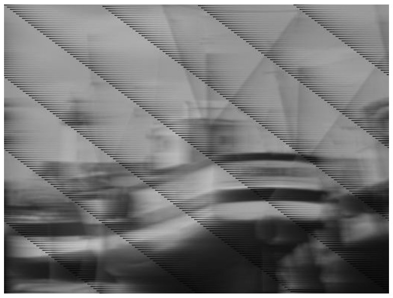


which is the step picture returned to life!

Let's try this unblurring on a real picture. Here is the example from lecture:



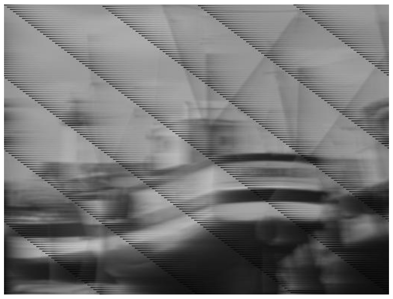


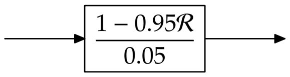


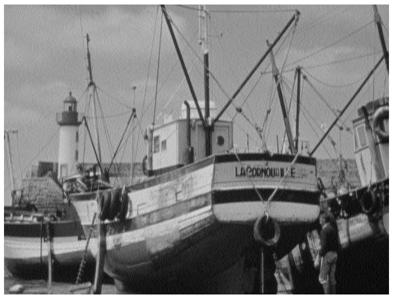
original

blurred

The situation looks dire. What hope is there of undoing such a drastic blurring? However, feeding the blurred picture through the unblurring system accurately reconstructs the original image:



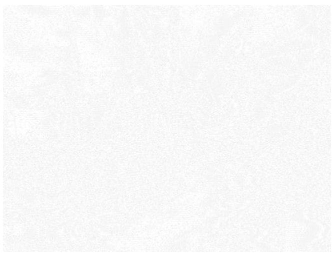




blurred

unblurred

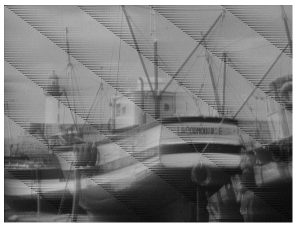
To determine the accuracy of the reconstruction, subtract the original image from the unblurred image and then (to save toner) invert the result. The result is shown in the margin figure. You almost certainly cannot see it in print, because it is light to begin with, and it passes through the laser-printer functional and the (even worse) photocopier functional.

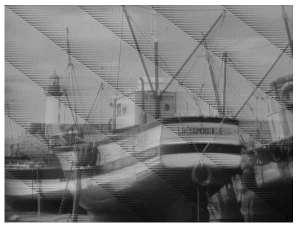


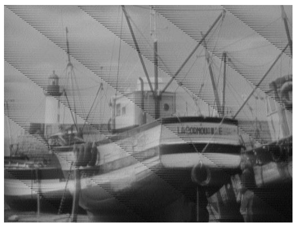
To minimize those problems, view the document online. You'll see that error is not zero. However, our theory based on pole–zero cancellation correctly unblurred the simple line images! The problem in the real-world image, represented in a typical image format, is quantization error. The simple line images used gray values represented as 32-bit floating-point numbers. In the real images, the gray values are 8-bit integers. A system that blurs across a wide region, which the system with a pole at 0.95 does, invites a large quantization error. In the error image, the maximum pixel value is 20, which is almost 10% of the range of pixel values (their range is $[0, 255]$).

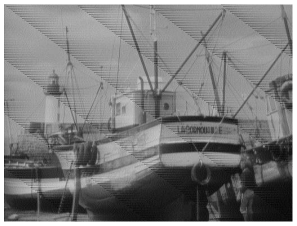
9.3 Mismatched unblurring

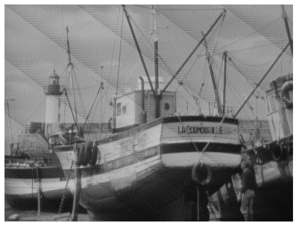
What if you do not know what the blurring system is? For example, it may be a picture taken last week with an out-of-focus digital camera, and now the camera is off and the week-old position of the lens is a mystery. This case is typical and makes reconstruction difficult. For example, here is a picture blurred with a mystery system. Let's say that you know the form of the system, for example, that it is a single pole somewhere between 0.9 and 0.98. Then try unblurring with a zero at 0.90, at 0.91, at 0.92, and so on until you find a good reconstruction. Here are several reconstructions:

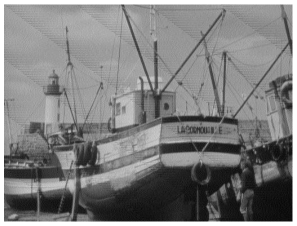

zero at 0.90

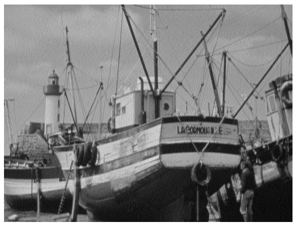

zero at 0.91

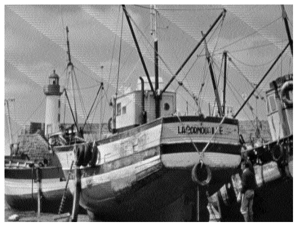

zero at 0.92

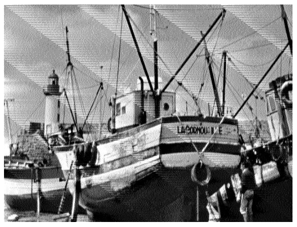

zero at 0.93


zero at 0.94


zero at 0.95


zero at 0.96

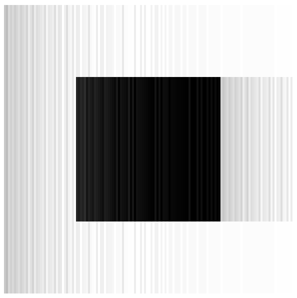

zero at 0.97

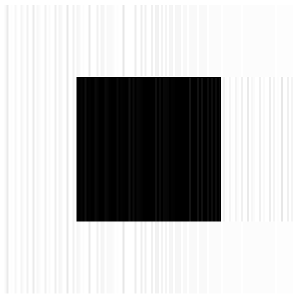

zero at 0.98

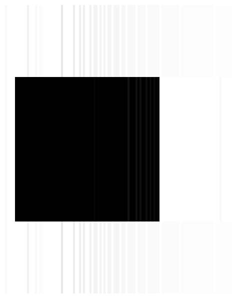
Putting a zero at 0.95, 0.96, and 0.97 all look reasonably sharp, with perhaps 0.96 being the first choice because it has no diagonal bands.

Exercise 53. What causes the diagonal bands? (I don't know but would like to.)

To decide among these choices, use a simple test image, which will be a black square on a white background. Blurring it with the mystery system and then unblurring it with various zeros produces:


zero at 0.95


zero at 0.96


zero at 0.97

The zero at 0.95 produces significant ghosting whereas the other two choices for the zero produce much less ghosting. So 0.96 and 0.97 look reasonable. In fact, the mystery system had a pole at 0.96, and the light banding in the correct reconstruction is due to quantization error.

Let's use operators to analyze the results of mismatching the pole and zero. If the mystery system's pole is at 0.96, then the system (including the gain) is

$$\frac{0.04}{1-0.96\mathcal{R}}.$$

A zero at 0.95, including gain, is the system $(1-0.95\mathcal{R})/0.05$. The combination is

$$\frac{0.04}{1-0.96\mathcal{R}} \times \frac{1-0.95\mathcal{R}}{0.05} = 0.8\,\frac{1-0.95\mathcal{R}}{1-0.96\mathcal{R}}.$$

The fraction can be simplified by subtracting and adding $0.01\mathcal{R}$ in the numerator, in order to take out the big part:

$$\frac{1-0.95\mathcal{R}}{1-0.96\mathcal{R}} = \frac{(1-0.95\mathcal{R}-0.01\mathcal{R})+0.01\mathcal{R}}{1-0.96\mathcal{R}}.$$

The parenthesized part of the numerator produces 1 (the big part), so the fraction is

$$1+\frac{0.01\mathcal{R}}{1-0.96\mathcal{R}}.$$

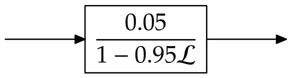
Now include the gain of 0.8. The combined pole–zero system is

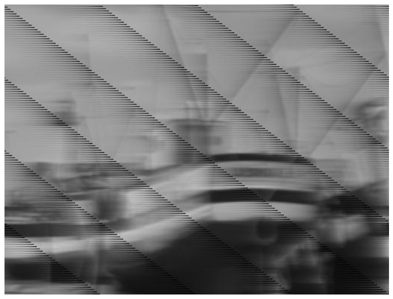
$$0.8+\frac{0.008\mathcal{R}}{1-0.96\mathcal{R}}.$$

The 0.8 produces a copy of the input image with slightly reduced intensities. The second term is a blurring system, though fortunately it contributes only a weak signal because of the 0.008 in the numerator. The original blurring system is $0.04/(1-0.96\mathcal{R})$, so in comparison this term is one-fifth of the original blurring system (with a single-pixel shift). The zero, although mismatched, cancels most (80%) of the blur. This pattern is not specific to image processing. If you have a feedback-control system with an unwanted pole, a reliable solution is to insert a nearby zero in the controller. As the zero approaches the pole, the cancellation becomes complete.

9.4 Causal and anticausal processing

The preceding examples used a causal system: one that uses no $\mathcal{L}$. The analysis is identical using an anticausal system: one that uses no $\mathcal{R}$, such as $0.05/(1-0.95\mathcal{L})$. Even the pictures look the same. For example, here is the boat picture processed through the leftward blur:

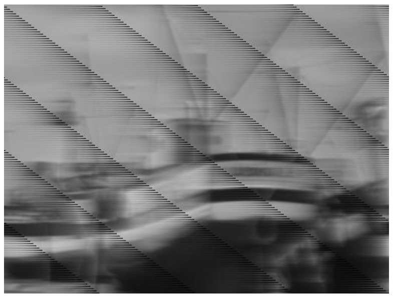




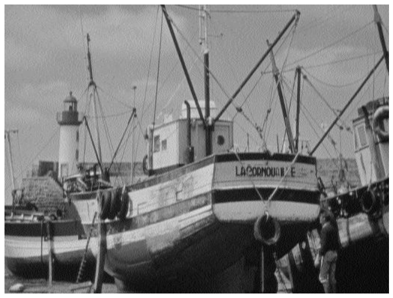
original

left blurred

And here is the result of unblurring it:



$$\begin{tikzcd} {} \arrow[r] & \boxed{\dfrac{1-0.95\mathcal{L}}{0.05}} \arrow[r] & {} \end{tikzcd}$$

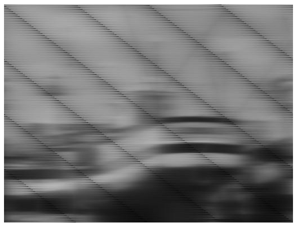


left blurred

left unblurred

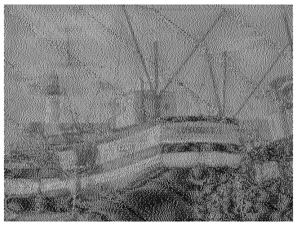
Because space has no preferred direction, a realistic blurring system does not prefer rightward (causal) or leftward (anticausal) blurs. It would instead combine the causal and anticausal systems to make a noncausal system. The combination system is a cascade of the causal and anticausal pieces, and has this effect:

$$\begin{tikzcd} {} \arrow[r] & \underset{\text{causal}}{\boxed{\dfrac{0.05}{1-0.95\mathcal{R}}}} \arrow[r] & \underset{\text{anticausal}}{\boxed{\dfrac{0.05}{1-0.95\mathcal{L}}}} \arrow[r] & {} \end{tikzcd}$$



To invert this blurring operation, reverse the flow direction: send the blurred image through the correct leftward zero and the correct rightward zero:

$$\begin{tikzcd} {} \arrow[r] & \underset{\text{anticausal}}{\boxed{\dfrac{1-0.95\mathcal{L}}{0.05}}} \arrow[r] & \underset{\text{causal}}{\boxed{\dfrac{1-0.95\mathcal{R}}{0.05}}} \arrow[r] & {} \end{tikzcd}$$



The reconstruction is terrible!

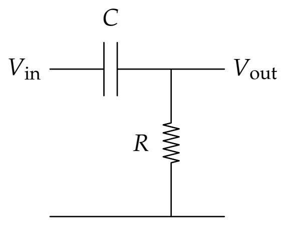
Exercise 54. [difficult!] What happened? How could you debug the problem and how might you fix it? There are at least two methods: the hack method and a principled method, but no fix can solve it completely.

Recitation 10
The integration operator

After studying this chapter, you should be able to:

- represent a first-order physical systems as a system functional using the integration operator $\mathcal{A}$, and call $\mathcal{A}$ your friend;
- expand first-order system functionals in a power series in $\mathcal{A}$ to compute output signals term by term; and
- recognize analogies between the discrete-time and continuous-time operator representations.

The preceding chapters used discrete-time systems to introduce several fundamental concepts of signals and systems: modes, feedback, and control. Discrete-time systems arise either by approximating continuous-time systems, perhaps using a forward- or backward-Euler approximation; or by construction, as in a simulation where time changes when and how you choose. All other systems are continuous-time systems because time is a continuous variable in the laws of physics. So we now use our experience with discrete-time systems to develop tools for continuous-time systems.



In analyzing discrete-time systems, we represented systems by their system functional constructed from the shift operators $\mathcal{R}$ and $\mathcal{L}$. Similarly, we represent continuous-time systems using an operator: the integration operator $\mathcal{A}$. To illustrate this operator, we study a *CR* circuit. Before analyzing it using operators, apply John Wheeler's maxim: Never make a calculation until you know the answer. First understand how the circuit behaves, then calculate! One way to understand a circuit is to think about similar circuits. A familiar circuit with the same two elements is the *RC* circuit. Compared to the *RC* circuit, the *CR* circuit swaps the elements. But swapping the R and the C does not change the current flowing from the V_{in} terminal to ground: The (complex) impedance of the R plus C combination is the sum of the individual impedances, whose order does not affect the sum. So the *RC* and *CR* circuits have the same current for the same V_{in}. Their only difference is the location of the output signal. In the *RC* circuit, the output signal is the voltage V_{C} across the capacitor. In the *CR* circuit, it is the voltage V_{R} across the resistor.

Exercise 55. In the *RC* arrangement (where the capacitor is connected to ground), what is the danger in measuring the capacitor voltage directly?

Because energy is conserved – in other words, the voltage around a loop is zero – the R and C voltages add to V_{in}:

$$V_{\text{out}}^{CR\text{ circuit}} + V_{\text{out}}^{RC\text{ circuit}} = V_{\text{in}}.$$

Thus we can compute the output signal for the CR circuit by computing it for the RC circuit and subtracting the result from the input signal.

For example, when the input signal is a step function V_0 (for $t \geq 0$), the output signal is

$$V_{\text{out}} = V_0\left(1 - e^{-t/\tau}\right) \qquad (RC \text{ circuit}),$$

where the time constant τ is the product RC. So the output signal of the CR circuit is V_0 minus the RC's output signal:

$$V_{\text{out}} = V_0 - V_0\left(1 - e^{-t/\tau}\right) = V_0 e^{-t/\tau} \qquad (CR \text{ circuit}).$$

The output is a decaying exponential.

We now check this result using operators. Once we are sure that the two methods give the same result, which increases our confidence in each method, we apply the operator method to a more complicated input signal than the step function.

10.1 Finding the system functional

Just as in discrete time, the first step in the operator method is to find the system functional. There are two ways to write a system functional for continuous-time systems: using differentiation or using integration operators. Differentiation operators seem easier because physical systems are often offered as differential equations. However, differentiation is a nasty operation that magnifies even trivial noise. For example, take a decent, well-behaved signal like a voltage increasing at a constant rate of 1 V/s. Now add a fluctuation of 1 nV for a very short time, say 1 ps. This tiny fluctuation contributes a huge derivative, on the order of 10^3 V/s. This misfeature of derivatives explains the Weierstrass function: a function that is continuous everywhere but has no derivative anywhere!

Exercise 56. [Difficult!] Construct a Weierstrass function.

Mathematicians had understood that a continuous function with kinks would have no derivative at the kinks. But a continuous function with no derivative anywhere shocked the mathematical community. For if that function had existed undetected for so long – allowing the community to think that a continuous function has a derivative except at a countable number of spots – then what other monsters lurked in the mathematical forest? So the Weierstrass function spurred a search for rigor to scrub the forest of demons.

If extended too long, this quest produces *rigor mortis*. In engineering and physics, where our models of the world are already approximate, we recommend a different philosophy: Don't worry, be happy. Do not agitate too much about rigor and avoid dangerous operations when possible.

Differentiation is such an operation, and physical systems do not differentiate. Therefore, we represent systems using the integration operator $\mathcal{A}$. It is defined by what it does to an input signal $f(t)$

$$\mathcal{A}f(t) = \int_{-\infty}^{t} f(t)\,dt.$$

Now we formulate the *CR* circuit using the $\mathcal{A}$ operator. A natural method is to use integral quantities, such as the charge Q on the capacitor. The voltage on the capacitor is $V_C = Q/C$, and the charge is the integral of the current:

$$V_C = \frac{1}{C}\int_{-\infty}^{t} I\,dt.$$

The resistor voltage is V_{out} and is also IR by Ohm's law. So

$$V_C = \frac{1}{RC}\int_{-\infty}^{t} V_{out}\,dt = \frac{1}{\tau}\mathcal{A}V_{out}.$$

Now write V_C in terms of the input and output voltages using conservation of energy:

$$V_C = V_{in} - V_{out}.$$

So

$$V_{in} - V_{out} = \frac{1}{\tau}\mathcal{A}V_{out}$$

or

$$\frac{V_{out}}{V_{in}} = \frac{1}{1+\frac{\mathcal{A}}{\tau}}.$$

Before using this (or any!) result, check its dimensions. The integration operator $\mathcal{A}$ has dimensions of time because it multiplies the function it acts on by dt, which is a (very tiny) time. So the ratio $\mathcal{A}/\tau$ is dimensionless. Thus the denominator $1 + \mathcal{A}/\tau$ makes dimensional sense and is dimensionless. The functional V_{in}/V_{out} is therefore dimensionless, as it should be being the ratio (loosely speaking) of two voltages V_{in} and V_{out}.

10.2 Step-function input

In discrete-time systems, the system functional is a recipe for finding how the system responds to input signals. The system functional $1 + 3\mathcal{R} + 2\mathcal{R}^2$, for example, turns the impulse into the signal $1, 3, 2, 0, 0, 0, \ldots$, a result that you obtain from the coefficients of the powers of $\mathcal{R}$. System functionals such as $1/(1 - 2\mathcal{R})$ first need to be expanded into the power series

$$1 + 2\mathcal{R} + 4\mathcal{R}^2 + 8\mathcal{R}^3 + \cdots.$$

Then you read off the impulse response as for a finite-polynomial system functional, and here it is $1, 2, 4, 8, \ldots$.

In continuous-time systems, the power-series expansion in the integration operator $\mathcal{A}$ provides an analogous recipe for finding output signals. No comparable recipe exists if differentiation is the fundamental operation. Imagine a power series in $\mathcal{D}$ applied to the step function. The first $\mathcal{D}$ produces a delta function, which is borderline bad news. And higher powers of $\mathcal{D}$ mean differentiating the delta function. What a horror! Each differentiation would magnify the infinity of the delta function, assuming that we even have permission to differentiate a delta function, and even worse to differentiate it multiple times. The absence of a meaningful power series in D is one more reason to select integration as the fundamental operation in continuous-time systems.

Let's take advantage of that choice and expand the *CR* circuit's system functional:

$$\begin{aligned}\frac{V_{\text{out}}}{V_{\text{in}}} &= \frac{1}{1+\frac{\mathcal{A}}{\tau}}\\ &= 1-\frac{A}{\tau}+\frac{A^2}{\tau^2}-\frac{A^3}{\tau^3}+\cdots.\end{aligned}$$

To compute the output signal, compute the effect of each term $(-A)^n/\tau^n$ on the input signal and add the results.

With the step function V_0 (for $t \geq 0$) as the input signal, the terms produce:

$$\begin{aligned}1V_{\text{in}} &= V_0,\\ -\frac{A}{\tau}V_{\text{in}} &= -V_0\frac{t}{\tau},\\ \frac{A^2}{\tau^2}V_{\text{in}} &= V_0\frac{1}{2!}\frac{t^2}{\tau^2},\\ -\frac{A^3}{\tau^3}V_{\text{in}} &= -V_0\frac{1}{3!}\frac{t^3}{\tau^3}.\\ &\cdots\end{aligned}$$

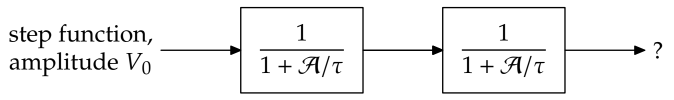
The sum is

$$V_{\text{out}} = V_0\left(1-\frac{t}{\tau}+\frac{1}{2!}\frac{t^2}{\tau^2}-\frac{1}{3!}\frac{t^3}{\tau^3}+\frac{1}{4!}\frac{t^4}{\tau^4}-\cdots\right) = V_0e^{-t/\tau}.$$

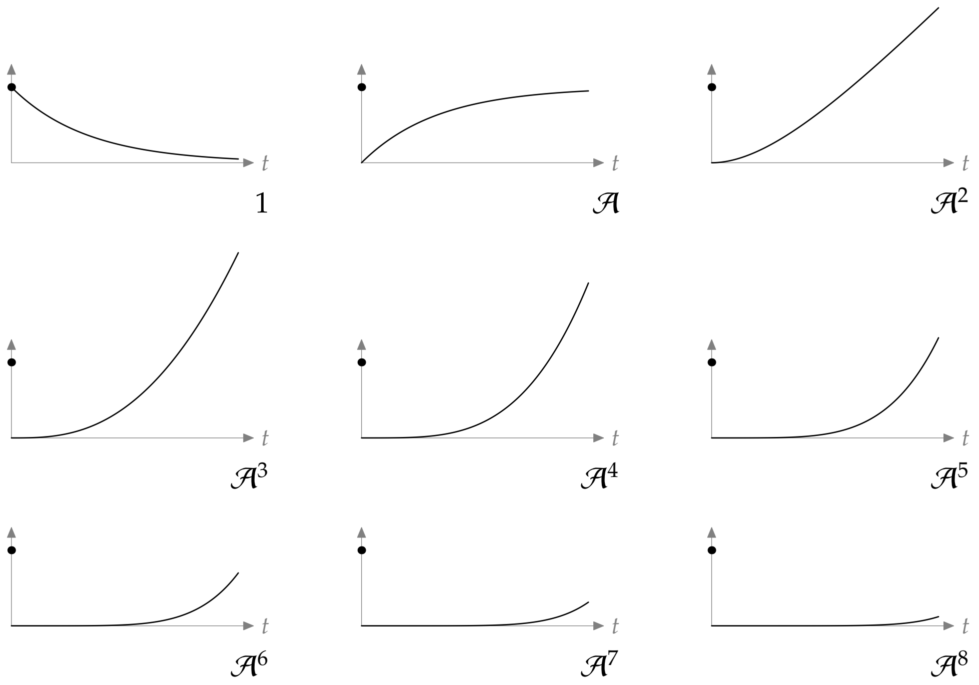
This result matches the result from taking the *RC* circuit's output and subtracting it from the input signal, when we applied Wheeler's maxim. So both results are likely to be correct. That confidence warrants trying a more complicated input signal.

10.3 Decaying exponential input

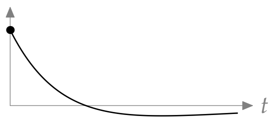
To prepare for making cascades of *CR* circuits, let's try feeding the circuit a decaying exponential $V_0e^{-t/\tau}$, which could be the output of an identical preceding *CR* circuit:


step function, amplitude V_0
$\frac{1}{1+\mathcal{A}/\tau}$
$\frac{1}{1+\mathcal{A}/\tau}$
?

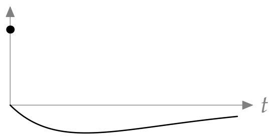
When we find modes of second-order continuous-time systems in the next chapter, we can use operators to analyze the cascade and to find the output of the second *CR* circuit when the first one is fed a step function. For now, let's use the first circuit's output that we computed in the preceding section, and find the output of the second directly by applying powers of $\mathcal{A}$ using numerical integration. To simplify the computations, set $\tau = 1$ and $V_0 = 1$. These choices are, in one way of looking at it, dimensionally unsound since neither time nor voltage are dimensionless. In another way of looking at it, these choices just mean that we agree to measure time in units of τ and voltage in units of V_0. Then the series to evaluate is $1 - \mathcal{A} + \mathcal{A}^2 - \cdots$ applied to the signal e^{-t}. The next graphs show the result of applying various powers of $\mathcal{A}$ to e^{-t}. They were computed using numerical integration. In each graph, the output value 1 is marked with a dot on the vertical axis.



Exercise 57. Check the first row by computing $\mathcal{A}e^{-t}$ and $\mathcal{A}^2e^{-t}$ analytically and sketching the results.



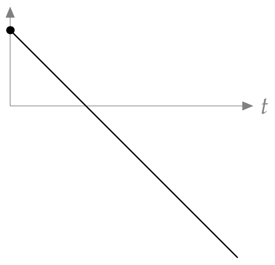
Combining these signals with alternating signs produces the output signal shown in the figure. Let's see whether we can guess a closed form for it by using approximation methods. The most useful technique for guessing function is to take out the big part. The signal looks like an exponential decay, which is reasonable at least in retrospect, since integrating e^{-t} produces another e^{-t}. So the output signal probably contains an e^{-t}, either as a separate term or as a factor multiplying other terms. Since both possibilities are reasonable, try both.



First assume that the e^{-t} is added to other stuff. So the output signal looks like

$$V_{\text{out}}(t) = Ce^{-t} + \text{stuff},$$

where we need to guess the constant C. Since the output signal starts at 1, a reasonable guess is that $C = 1$. Then the other stuff starts at zero, which may simplify it. To guess the form of the stuff, subtract e^{-t} from the output signal. The result, computed numerically again, is shown in the figure. Its shape looks familiar and could be $-te^{-t}$, which is also linear near the origin and decays to zero for large t.



If that guess is correct, then the alternative way to take out the big part, by removing a factor of e^{-t}, should work well. Factoring out e^{-t} from the output signal produces the function in the margin (computed numerically). That function is easier to guess than is the preceding curve: It is $1 - t$. So the output signal is

$$V_{\text{out}}(t) = (1 - t)\, e^{-t}.$$

Exercise 58. Confirm this result by setting up the differential equation and solving it using what your knowledge from 18.03 (or otherwise).

You can solve the differential equation analytically because the input signal e^{-t} is a friendly function. If the input signal were $1/(1+t^4)$, a closed-form solution would be more difficult if not impossible. However, by combining the operator expansion with numerical integration, you can compute output signals even when the input signal is unfriendly and has no closed-form integral. In practice, this integration series is numerically expensive, and the more efficient method is to turn the differential equation into a discrete-time system using the forward- or backward-Euler approximations. The integration series does, however, provide a global, whole-signal picture of how the output signal becomes the input signal, in contrast to the sample-by-sample picture provided by using a discrete-time approximation of the differential equation. The integration series also provides a way to construct some closed-form solutions without guessing them ahead of time – as long as you recognize the power series for e^t in various disguises.

The next chapter takes this experience with $\mathcal{A}$ and applies it to second-order continuous-time systems, which are the important systems in engineering.

Recitation 11
Oscillating double poles

After studying this chapter, you should be able to:

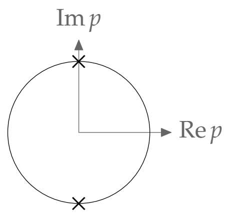
- do a partial-fractions decomposition for system functionals with double poles;
- use the partial-fractions expansion to compute the impulse response; and
- appreciate the physical consequences of resonance.

In this chapter, we use the $\mathcal{A}$ operator to study higher-order systems. The example is a cascade of two identical mass–spring systems:

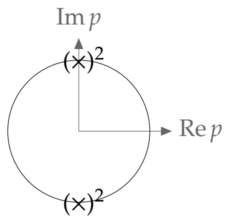
$$\begin{tikzcd}
\delta \arrow[r] & \boxed{\dfrac{\mathcal{A}^2}{1+\mathcal{A}^2}} \arrow[r] & \boxed{\dfrac{\mathcal{A}^2}{1+\mathcal{A}^2}} \arrow[r] & \text{impulse response of the cascade} \\
& \text{mass–spring} & \text{mass–spring} &
\end{tikzcd}$$

Each mass–spring subsystem has two poles, so the cascade has four poles. Four poles looks formidable! However, by subdividing the problem into known problems – two pairs of double poles – you can understand how this complex system behaves. As a side benefit, we begin investigating resonance, a ubiquitous phenomena that we return to when we study signal processing.

11.1 Factored form



The problem is specified as a cascade of two subsystems with a known factorization, so the factored form of the system functional is easy to generate. Each mass–spring subsystem has system functional $\mathcal{A}^2/(1+\mathcal{A}^2)$. The full functional is $\omega_0^2\mathcal{A}^2/(1+\omega_0^2\mathcal{A}^2)$, where $\omega_0 \equiv \sqrt{k/m}$ and k is the spring constant and m is the mass. For notational simplicity we measure time relative to one radian of oscillation, so we use time units of $1/\omega_0 = \sqrt{m/k}$. Equivalently, we choose units such that $m = 1$ and $k = 1$. Then $\omega_0 = 1$. With this simplification, the poles of each subsystem lie on the unit circle. The pole–zero diagram for one mass–spring subsystem is shown in the margin.



The cascade of two mass–spring subsystems has a system functional that is the square of the single subsystem's functional, so it is

$$\left(\frac{\mathcal{A}^2}{1+\mathcal{A}^2}\right)^2.$$

Squaring a functional duplicates each pole (and zero), so the whole system's pole–zero diagram is the pole–zero diagram of a single mass–spring subsystem but with each pole doubled in multiplicity. The same doubling would happen to the zeros, but this system has no zeros.

From the system functional we can find the impulse response. But first follow Wheeler's maxim: Never calculate until you know the answer already! As shown in lecture, and by analogy with discrete-time double poles, these double poles at $\pm j$ produce responses that contain $t\cos t$, $t\sin t$, $\cos t$, and $\sin t$. The weights on these terms depend on the input signal. The factored form, which is a series decomposition, does not directly tell us the weights. To find the weights, we use the factored form to do a parallel decomposition. Each term in the parallel decomposition produces one mode in the impulse response. The decomposition might be a lengthy calculation, but the rough analysis at the start of this paragraph tells us what to expect.

11.2 Modal decomposition

A parallel decomposition means a partial-fractions decomposition of the system functional. The direct approach is to write the partial-fraction terms and to solve for the unknown weights. As in discrete-time systems, the number of terms is the number of poles. So we expect four terms.

Also as in discrete-time systems, each double pole contributes two terms. If the double pole is at p, its two contributions are

$$\frac{p\mathcal{A}}{1-p\mathcal{A}} \quad \text{and} \quad \left(\frac{p\mathcal{A}}{1-p\mathcal{A}}\right)^2.$$

The double poles of the cascade are at $p = \pm j$, so the four partial-fraction terms are

$$\frac{\mathcal{A}}{1-j\mathcal{A}}, \quad \frac{\mathcal{A}}{1+j\mathcal{A}}, \quad \frac{\mathcal{A}^2}{(1-j\mathcal{A})^2}, \quad \text{and} \quad \frac{\mathcal{A}^2}{(1+j\mathcal{A})^2},$$

where the factors of j have been absorbed into the unknown weights. We have to chose the weights so that the weighted sum produces the original system functional, and making this choice requires solving four equations in four unknowns. Finding the weights is messy.

Instead we decompose using a shortcut. The long method that we avoid involves squaring the single mass–spring subsystem functional and then decomposing the four-pole system into partial fractions. Why not reverse the order? To do so, expand one mass–spring subsystem into partial fractions, *then* square the decomposition. The general principle behind this reversal is to keep expressions as organized as possible: to keep them in a low-entropy form. Decomposing one subsystem requires finding only two weights, which is an easier and more organized calculation than is finding four weights. And squaring a known function maintains an organized form, turning the square of two terms into three terms with no unknown coefficients.

The first step, then, is to decompose one mass–spring subsystem. It has poles at $\pm j$, so it decomposes like so:

$$\frac{\mathcal{A}^2}{1+\mathcal{A}^2} = a\frac{\mathcal{A}}{1+j\mathcal{A}} + b\frac{\mathcal{A}}{1-j\mathcal{A}}.$$

To make the numerator turn into $\mathcal{A}^2$ after adding the fractions, choose $a = j/2$ and $b = -j/2$. Then the decomposition is

$$\frac{\mathcal{A}^2}{1+\mathcal{A}^2} = \frac{1}{2}\left(\frac{j\mathcal{A}}{1+j\mathcal{A}} + \frac{-j\mathcal{A}}{1-j\mathcal{A}}\right).$$

Now square this decomposition to get a decomposition for the cascade:

$$\left(\frac{\mathcal{A}^2}{1+\mathcal{A}^2}\right)^2 = \frac{1}{4}\left(\frac{j\mathcal{A}}{1+j\mathcal{A}} + \frac{-j\mathcal{A}}{1-j\mathcal{A}}\right)^2.$$

Squaring the sum in parentheses produces three terms:

$$\frac{j^2\mathcal{A}^2}{(1+j\mathcal{A})^2}, \quad \frac{2\mathcal{A}^2}{1+\mathcal{A}^2} \quad \text{and} \quad \frac{(-j)^2\mathcal{A}^2}{(1-j\mathcal{A})^2}.$$

So a semi-complete partial-fractions decomposition is (after changing $(-j)^2$ to j^2 and vice versa):

$$\frac{1}{4}\underbrace{\frac{(-j)^2\mathcal{A}^2}{(1+j\mathcal{A})^2}}_{\textit{double pole at } -j} + \frac{1}{2}\underbrace{\frac{\mathcal{A}^2}{1+\mathcal{A}^2}}_{\textit{mass–spring subsystem}} + \frac{1}{4}\underbrace{\frac{j^2\mathcal{A}^2}{(1-j\mathcal{A})^2}}_{\textit{double pole at } j}.$$

The decomposition is incomplete because the middle term can be further expanded. However, there is no need to expand it. Remember that the goal of the partial-fractions expansion was not algebra gymnastics, but rather the impulse response. The algebra is a means to an end. And we know the impulse response of the middle term because, except for the factor of 1/2, the middle term is the system functional of one mass–spring subsystem. We can find its impulse response either from solving the mass–spring differential equation or by using the result from lecture, when we decomposed it into partial fractions and found its impulse response directly. So there is no need to repeat the calculation.

The other two terms have the double-pole form

$$\left(\frac{p\mathcal{A}}{1-p\mathcal{A}}\right)^2,$$

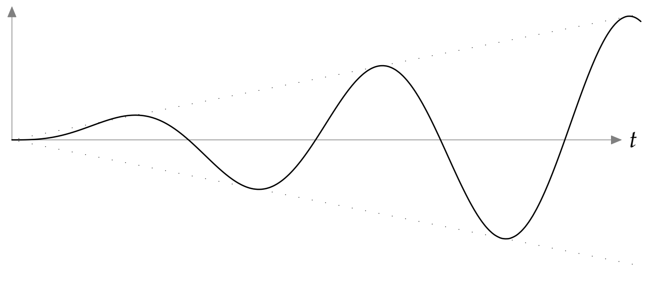
where one term has $p = j$ and the other has $p = -j$. In lecture we calculated the impulse response of this double-pole form, and found that the response is p^2te^{pt}. After including the factor of 1/4, the two terms together contribute these terms to the impulse response:

$$\frac{1}{4}\left(-te^{jt} - te^{-jt}\right) = -\frac{1}{2}t\cos t,$$

where we used Euler's formula $\cos t = (e^{jt} - e^{-jt})/2$. The middle, mass–spring term, including the factor of 1/2, contributes $(\sin t)/2$. So the combined impulse response of the cascade is

$$\frac{\sin t - t\cos t}{2}.$$

Here is its sketch including dotted $\pm t$ asymptotes:



Let's check this result by looking first at the extreme case $t \to 0$.

Exercise 59. Take the $t \to 0$ limit of the impulse response to show that the impulse response is $t^3/6$ near the origin.

When t is small, the impulse response predicts $t^3/6$, which we can confirm with the following physical argument. The position impulse on the first subsystem gives its mass a force impulse (by Hooke's law). A force impulse is like hitting the mass with a fast hammer. The unit force impulse, acting on a unit mass, gives the mass an initial velocity of 1. The position of the mass therefore is the integral of the velocity, so the position is t. This position is the input signal for the second mass–spring subsystem.

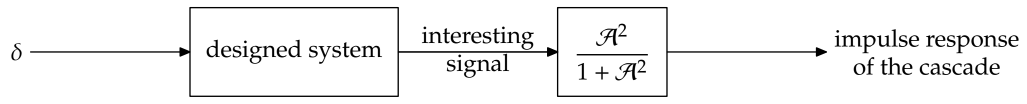
Let's see what that position does to the second subsystem. A linearly growing input signal provides, by Hooke's law, a linearly growing force on the mass. This force produces a linearly growing acceleration t (the mass is 1 in these units). So the velocity is $t^2/2$, and the position is $t^3/6$, which is the prediction based on the impulse response that we computed! This confirmation in the $t \to 0$ limit increases our confidence in the whole impulse response $(\sin t - t\cos t)/2$.

This analysis applied for small t. As t grows, the masses move significantly, changing the stretch in the spring and changing the forces. Those effects are accounted for by the feedback loops in the block diagram. When $t \approx 0$, however, the signals do not have time (speaking roughly) to go round the feedback loop. Then the output signal is the result of sending the input signal straight through the forward paths. The forward path through the cascade is $\mathcal{A}^4$, and indeed $\mathcal{A}^4\delta(t) = t^3/6$. So we have a mathematical confirmation of the physical argument.

11.3 Resonance

Now let's look at the large-t behavior of the output. The $\sin t$ term is not important compared to $t\cos t$. So the output oscillates with an amplitude that becomes infinite, as shown in the preceding sketch. You can see why this result makes sense by using a physical argument. The first mass–spring subsystem produces the output signal $\sin t$, which is an oscillation with angular frequency 1 (remember that $\omega_0 = 1$). This output drives the second mass–spring subsystem, which has a natural (angular) frequency of 1. So the intermediate signal drives the second subsystem at its natural frequency. It is like pushing an undamped swing at its natural frequency. The oscillations grow and grow until the swing set breaks. This failure is a nonlinearity, which is beyond the scope of this course, so in our analysis the swing's amplitude would grow without bound as we feed energy into it.

This whole analysis is one way to understand signal processing before studying Fourier series and the so-called Fourier transform. For the moment, we can compute impulse responses, and not much else. So how to understand how a mass–spring system responds to an interesting input signal? Make an interesting signal by designing a system whose impulse response is interesting. Then feed that signal to the mass–spring subsystem:


δ
designed system
interesting signal
$\frac{\mathcal{A}^2}{1+\mathcal{A}^2}$
impulse response of the cascade

The final output is the impulse response of the cascade. If we can compute the impulse response of the cascade, which usually starts by factoring its system functional and expanding in partial fractions, then we can find the interesting-signal response of the second subsystem.

This chapter illustrated a special case of that method. To find how an oscillating input – the interesting signal – affects the mass–spring subsystem, feed an impulse into one mass–spring subsystem to produce the oscillating signal. Then feed that intermediate signal into the second mass–spring subsystem. So we end up analyzing the impulse response of the cascade of two mass–spring subsystems. Because the subsystems are identical, the oscillating output of the first subsystem drives the second system at resonance, so the final output should (and does) oscillate to infinity.

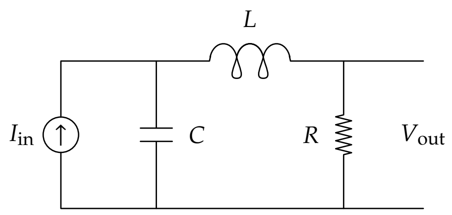
Exercise 60. *[slightly unfair but worth thinking about]* Linear systems, when given a pure oscillating input, are supposed to produce an output signal that is also a pure oscillating output and at the same frequency. However, the oscillating input to the second subsystem produced a funny signal: $t\cos t$. That signal is not a single frequency. What happened? Is linear-systems theory wrong?

Recitation 12
State variables

After studying this chapter, you should be able to:

- identify the state variables in an *LRC* circuit;
- use state variables to write the corresponding differential equation;
- predict the impulse response for extreme values of the quality factor Q; and
- estimate Q from the impulse response.

Here is an *LRC* circuit driven by a current source:



We analyze it in four steps. First, we find the state variables. Second, we use the state variables to set up a differential equation. Third, we convert the differential equation into a system functional. Fourth, we use the functional to find the modes and to understand the impulse response in the extreme cases of damping (little damping and lots of damping).

12.1 Finding state variables

Pause to try 46. What are the state variables of the circuit?

State variables are the minimum knowledge of the past needed to propagate the output into the future. Let's apply that definition one by one to the fundamental circuit elements L, R, and C. A resistor is the simplest circuit element, so start with it.

Pause to try 47. What is the state variable for a resistor?

Thanks to Ohm's law, if you know the current through a resistor, you know the voltage without needing to know the history of the current. Conversely, if you know the voltage now, you know the current now without needing to know the history of the voltage. A resistor, therefore, has no memory and therefore no state variables!

Pause to try 48. What is the state variable for an inductor?

To decide the state variable, look at what an inductor does. An inductor is a magnetic element that relates voltage to the changing magnetic flux produced by a changing current:

$$v_L = L\frac{di_L}{dt}.$$

There are two possibilities for the inductor's state variable: v_L or i_L. The relation differentiates the current, so knowing the voltage does not constrain the constant of integration. And it is that constant of integration where the state lives. Therefore knowing the voltage is incomplete information. Instead we would rather know the inductor current, which means knowing the integral of the voltage. So current i_L is the state variable for the inductor.

Exercise 61. Could the magnetic flux in the inductor be a state variable?

Pause to try 49. What is the state variable for a capacitor?

Capacitors and inductors are duals of one another. If current is the state variable for an inductor, voltage should be the state variable for a capacitor. Let's check that intuition with an equation. A capacitor is an electric element – as opposed to a magnetic element – that relates voltage to stored charge:

$$Q = Cv_C.$$

Its derivative gives the relation between current and voltage:

$$i_C = C\frac{dv_C}{dt}.$$

The current is not full information because, thanks to the derivative connecting it to v_C, it does not contain the constant of integration. The state variable is the voltage v_C or the charge Q. Those choices are proportional to one another, so it does not matter which one you choose. We use v_C.

12.2 Writing the differential equation

The next step is to use the state variables to write the differential equation. There are two state variables so we need two equations. The first comes from conservation of charge, also known as Kirchoff's current law (KCL). After the current leaves the current source, it reaches the first node and can go through the inductor or the capacitor. So

$$I_{\text{in}} = i_C + i_L.$$

The state variable for the capacitor is v_C, so in terms of the state variables, conservation of charge says that

$$I_{\text{in}} = C\frac{dv_C}{dt} + i_L.$$

Let's call this equation the charge equation.

The second equation comes from conservation of energy, also known as Kirchoff's voltage law, which says that the sum of the voltages around the loop is zero:

$$v_C = v_L + i_L R.$$

The inductor voltage is not a state variable. So rewrite the equation using the state variable i_L:

$$v_C = L\frac{di_L}{dt} + i_L R.$$

Let's call this equation the energy equation.

Now we combine the two equations into one differential equation. The charge equation

$$I_{\text{in}} = C\frac{dv_C}{dt} + i_L$$

is a good place to start because it contains the input signal I_{in}.

Pause to try 50. In finding the differential equation, should we eliminate v_C or i_L?

The output voltage is across the resistor, and the same current flows through the resistor and inductor (why?). So the output voltage is

$$V_{\text{out}} = i_L R,$$

suggesting that we should eliminate v_C rather than i_L. To eliminate v_C from the charge equation, use the energy equation

$$v_C = L\frac{di_L}{dt} + i_L R.$$

The charge equation contains dv_C/dt, so differentiate the energy equation to get

$$\frac{dv_C}{dt} = L\frac{d^2 i_L}{dt^2} + R\frac{di_L}{dt}.$$

The charge equation becomes

$$\begin{aligned} I_{\text{in}} &= C\frac{dv_C}{dt} + i_L \\ &= LC\frac{d^2 i_L}{dt^2} + RC\frac{di_L}{dt} + i_L. \end{aligned}$$

The output signal is $V_{\text{out}} = i_L R$, so in terms of the output signal, the equation is

$$RI_{\text{in}} = LC\frac{d^2 V_{\text{out}}}{dt^2} + RC\frac{dV_{\text{out}}}{dt} + V_{\text{out}}.$$

12.3 Finding the system functional

Finding the system functional is the next step. To do so, multiply the differential equation by enough powers of $\mathcal{A}$ to get rid of the differentiations. Here it means multiplying by $\mathcal{A}^2$, which gives

$$R\mathcal{A}^2 I_{\text{in}} = (LC + RC\mathcal{A} + A^2)V_{\text{out}}.$$

So the system functional is

$$\frac{V_{\text{out}}}{I_{\text{in}}} = \frac{R\mathcal{A}^2}{LC + RC\mathcal{A} + A^2}.$$

Using $\omega_0 = 1/\sqrt{LC}$, the functional is

$$\frac{V_{\text{out}}}{I_{\text{in}}} = R\frac{\omega_0^2\mathcal{A}^2}{1 + \frac{R}{L}\mathcal{A} + \omega_0^2\mathcal{A}^2}.$$

Pause to try 51. Check the dimensions.

Before analyzing this functional, let's check the dimensions. For example, in the denominator, the $\omega_0^2\mathcal{A}^2$ term has two powers of frequency from the ω_0^2 and two powers of time from the $\mathcal{A}^2$. So it is dimensionless, as it should be since it is being added to the 1, which is dimensionless.

Exercise 62. What are the dimensions of $(R/L)\mathcal{A}$?

The numerator $\omega_0^2\mathcal{A}^2$ is also dimensionless, so the only dimensional factor in the system functional is the prefactor R. The system functional is the ratio of current to voltage, so it is an impedance, so it needs a prefactor like R to give it the proper dimensions.

12.4 Parameterizing the behavior

This system functional, except for the R prefactor, is the one analyzed in Lecture 11. Rather than redo the derivation, we quote the main results, looking at whether the impulse response is reasonable, and sketch outputs for the extremes of damping.

The main idea is to measure time in the form $\tilde{t}$ where $\tilde{t} \equiv \omega_0 t$. Since ω_0 is a frequency, $\tilde{t}$ is dimensionless. Similarly, use $\widetilde{\mathcal{A}}$, which integrates with respect to $\tilde{t}$ instead of with respect to t.

Pause to try 52. What are the dimensions of $\widetilde{\mathcal{A}}$?

$$\widetilde{\mathcal{A}} = \int_{-\infty}^{t} (\cdot)\, d\tilde{t} = \omega_0 \underbrace{\int_{-\infty}^{t} (\cdot)\, dt}_{\mathcal{A}}.$$

So $\widetilde{\mathcal{A}} = \omega_0\mathcal{A}$. Since $\mathcal{A}$ has dimensions of time, $\widetilde{\mathcal{A}}$ is dimensionless.

Exercise 63. What is the system functional in terms of $\widetilde{\mathcal{A}}$?

In terms of $\widetilde{\mathcal{A}}$, the system functional is

$$\frac{V_{\text{out}}}{I_{\text{in}}} = R\frac{\widetilde{\mathcal{A}}^2}{1+\frac{R}{\omega_0 L}\widetilde{\mathcal{A}}+\widetilde{\mathcal{A}}^2}.$$

Except for the R prefactor, the only variables are in the dimensionless combination $R/\omega_0 L$. Using the dimensionless parameter $Q \equiv \omega_0 L/R$, the system functional is

$$\frac{V_{\text{out}}}{I_{\text{in}}} = R\frac{\widetilde{\mathcal{A}}^2}{1+\frac{1}{Q}\widetilde{\mathcal{A}}+\widetilde{\mathcal{A}}^2}.$$

The main result is that the system is characterized by three parameters:

1. the prefactor R, which is a scale factor between current and voltage;
2. the corner frequency ω_0, which is a scale factor for time; and
3. the dimensionless factor Q, the so-called quality factor.

The first two parameters are just scale factors. The interesting parameter is the Q because changing Q changes the behavior of the system qualitatively.

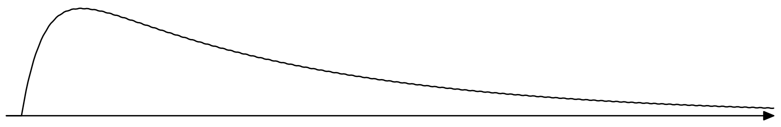
The conclusion is that the three parameters of the original system – L, R, and C – combine into one important parameter. This result is fundamentally important:

A good table of functions of one variable may require a page; that of a function of two variables a volume; that of a function of three variables a bookcase; and that of a function of four variables a library.

—Harold Jeffreys

12.4.1 The low-Q limit

Here is one impulse response:



Pause to try 53. Estimate the Q for the above shape.

What about the scales on the axes? The two other parameters R and ω_0 set the scales on the input and output axes, so we leave those axes unscaled to signify that we are not worrying about those parameters. You can determine the Q simply from the shape of the output without knowing the scales on the axes.

This non-oscillating shape suggests a damped system. Let's hope for strong damping because it's the easier to analyze than moderate damping. Strong damping means $Q \approx 0$. So let's figure out the impulse response in the $Q \to 0$ limit, and then match that analysis to the sketch. In the low-Q limit, the denominator is easy to factor:

$$1 + \frac{1}{Q}\widetilde{\mathcal{A}} + \widetilde{\mathcal{A}}^2 \approx (1 + \frac{1}{Q}\widetilde{\mathcal{A}})(1 + Q\widetilde{\mathcal{A}}).$$

Exercise 64. Check this factorization.

So the poles in the $\tilde{p}$ plane are $-1/Q$ and $-Q$, and the two modes are $e^{-Q\tilde{t}}$ and $e^{-\tilde{t}/Q}$. The impulse response is a linear combination of these modes. We can figure out the combination by thinking about how it must behave. At short times the system looks like double integration, because of the $\widetilde{\mathcal{A}}^2$ in the numerator. An impulse input becomes a step function after one integration and a ramp after a second integration. So the impulse response should grow linearly for $\tilde{t}$ near 0.

To satisfy that requirement, the impulse response subtracts the modes:

$$y(t) \propto e^{-Q\tilde{t}} - e^{-\tilde{t}/Q}.$$

To find the constant of proportionality, look at the area of the impulse response. The $e^{-\tilde{t}/Q}$ mode decays rapidly, so the area is mostly contributed by the slow-decay mode $e^{-Q\tilde{t}}$. It has area $1/Q$, which is large. To make the impulse response have unit area like the impulse has, we should multiply by Q:

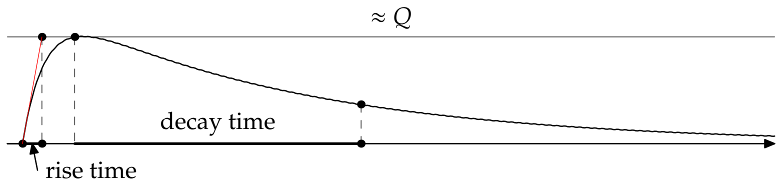
$$y(t) \propto Q\left(e^{-Q\tilde{t}} - e^{-\tilde{t}/Q}\right).$$

The proportionality symbol remains because the output variable may contain dimensional factors such as R (which it does in this example). But the dimensionless portion is the interesting part.

The response contains a fast-decay mode $e^{-\tilde{t}/Q}$ and a slow-decay mode $e^{-Q\tilde{t}}$. The maximum occurs once the fast-decay mode dies out and before the slow-decay mode has decayed significantly. When Q is very small, those time scales are very different, and the slow-decay mode has hardly changed at all even when the fast-decay mode is nearly wiped out. So, at the maximum, the slow-decay mode is still nearly 1, and the maximum output is therefore Q. How long does it take for the output to reach the maximum? An approximation to that (scaled) time we will call the rise time, which is the (scaled) time for the initial slope to hit the maximum line at Q. The slope near $\tilde{t} = 0$ is 1. Thus the initial slope intersects the maximum horizontal line at $\tilde{t} = Q$. So the (scaled) rise time is Q.

At large $\tilde{t}$, the fast-decay mode – also called the transient – has vanished, so the impulse response is $Qe^{-Q\tilde{t}}$. That decays by a factor of e in a scaled time $\Delta\tilde{t} = 1/Q$. So we can find Q from a sketch by taking the ratio of the rise time and the decay time.

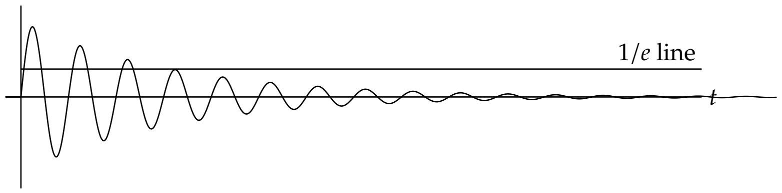
Here is that sketch with those times marked:



You can use a ruler to measure the rise and decay times. The figure-drawing program says that the rise time is 2.6 mm long and that the decay time is 38 mm long. Their ratio is 14.6 so the Q is approximately $1/\sqrt{14.6} \approx 0.25$. The value used to generate that curve is 0.3 so this approximate method provides a reasonable answer, especially considering that it is designed for $Q \to 0$.

12.4.2 High-Q limit

Here is an impulse response in the high-Q limit:



This response looks almost like the response of an undamped mass–spring system or of an LC circuit, which is also undamped. The undamped limit happens when the $\widetilde{\mathcal{A}}$ term vanishes from the denominator, which happens when $Q \to \infty$. So let's find the modes and the impulse response in this limit, and match the result to the sketch to find Q.

In the infinite-Q limit, the (scaled) poles would be at $\pm j$. So in the high-Q limit, they will be close to $\pm j$. Let ϵ be the small deviation. Then $\tilde{p} = \epsilon \pm j$. The product of the poles should be 1 because 1 is the coefficient of the $\widetilde{\mathcal{A}}^2$ term in the denominator. If we ignore the ϵ^2 term that results from multiplying these two poles, the product is still 1, so that's okay. The sum of the poles has to be $1/Q$ because $1/Q$ is the coefficient of $\widetilde{\mathcal{A}}$ in the denominator. So

$$(\epsilon + j) + (\epsilon - j) = \frac{1}{Q},$$

which means that $\epsilon = -1/2Q$. So the poles are

$$\tilde{p} \approx -\frac{1}{2Q} \pm j.$$

The imaginary component j makes the modes oscillate, and the tiny negative real part makes those oscillations decay. The modes are

$$e^{(j-1/2Q)\tilde{t}} \quad \text{and} \quad e^{(-j-1/2Q)\tilde{t}}.$$

The impulse response still starts from 0, so the impulse response subtracts these modes to get a zero difference at $\tilde{t} = 0$:

$$y(t) \propto e^{(j-1/2Q)\tilde{t}} - e^{(-j-1/2Q)\tilde{t}}.$$

Near $\tilde{t} = 0$ the impulse response is the ramp $y(t) \propto \tilde{t}$, by the same double-integration argument given for the low-Q limit. To make its slope be 1, we need to divide by $2j$. Therefore the impulse response is

$$y(t) \propto e^{-\tilde{t}/2Q} \sin \tilde{t}.$$

The Q shows up only in the exponential decay, and it makes the signal decay by a factor of e in a (scaled) time $2Q$. But if the time axis is not labelled, how can you tell what the time is? Because the oscillating factor provides a clock for the $\tilde{t}$ axis. The oscillation completes a full cycle when $\tilde{t}$ advances by 2π. So

$$2Q = \frac{\text{length on horizontal axis for amplitude to fall by } e}{\frac{1}{2\pi} \times \text{length on horizontal axis for one cycle}}$$

So

$$\begin{aligned} Q &= \pi \times \frac{\text{length on horizontal axis for amplitude to fall by } e}{\text{length on horizontal axis for one cycle}} \\ &= \pi \times \text{number of cycles for amplitude to fall by } e. \end{aligned}$$

The sketch looks like it takes a about one-quarter cycle plus 3 full oscillations to decay to the $1/e$ line. The one-quarter cycle is the time for the $\sin \tilde{t}$ to reach its first peak at $\tilde{t} = \pi/2$. So

$$Q \approx \pi \times 3.25 \,\text{cycles} \approx 10.2.$$

The value used to generate the curve was $Q = 10$, so this eyeball method is more accurate than we might expect.

Exercise 65. Sketch the (dimensionless) pole locations for $Q = 0.25$ and $Q = 10$.

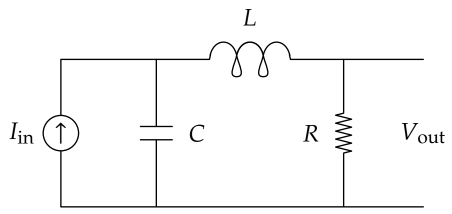
Exercise 66. Find the step response of this LRC system in the high- and low-Q limits.

Recitation 13
Eigenfunctions and frequency response

After studying this chapter, you should be able to:

- find the system function $H(s)$ for a system characterized by a second-order linear, constant-coefficient differential equations;
- explain the analogy between eigenvectors in finite-dimensional systems (plane geometry!) and eigenfunctions in our systems; and
- use $H(s)$ to sketch the frequency response, and estimate Q from a sketch of the frequency response.

We will illustrate a new representation, the system function, by analyzing the same system as in the last chapter:



In the last chapter we found its differential equation:

$$RI_{\text{in}} = LC\frac{d^2V_{\text{out}}}{dt^2} + RC\frac{dV_{\text{out}}}{dt} + V_{\text{out}}.$$

From the differential equation we can go either to the system functional using $\mathcal{A}$ and then to the system function, or directly to the system function. We'll go via the system functional, which is

$$\frac{V_{\text{out}}}{I_{\text{in}}} = R\frac{\omega_0^2\mathcal{A}^2}{1 + \frac{R}{L}\mathcal{A} + \omega_0^2\mathcal{A}^2}.$$

To get the system function, replace $\mathcal{A}$ by $1/s$. This replacement is analogous to the replacement of $\mathcal{R}$ with $1/z$ to get the discrete-time system function $H(z)$. Replacing $\mathcal{A}$ by $1/s$ gives

$$H(s) = R\frac{\omega_0^2/s^2}{1 + (R/L)/s + \omega_0^2/s^2}.$$

Using the definition $Q \equiv \omega_0 L/R$, it simplifies to

$$H(s) = R\frac{1}{1 + (s/\omega_0)/Q + s^2/\omega_0^2}.$$

The $H(s)$ is called the system function. The roots of its denominator give the poles, and the roots of its numerator (which is boring here) give the zeros. We have already studied the poles of the system functional, whose denominator has the same structure as the system function's denominator. Rather than redoing the factorization with s instead of $\mathcal{A}$, which is a minor change, we instead focus on eigenfunctions and frequency response, which are two new ideas brought by $H(s)$. To summarize what we will find, we choose special, oscillating functions – eigenfunctions – for which the system behaves simply. We then represent the system according to how it acts on those functions. This representation is the frequency response.

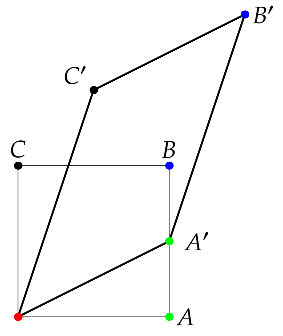
13.1 Eigenvectors and eigenfunctions

Before leaping into eigenfunctions, look at a simpler version of the same idea by taking an example from plane geometry. Here is a matrix that turns two-dimensional vectors into other two-dimensional vectors:

$$T = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 3/2 \end{pmatrix}.$$

The operator T is a system that transforms vectors. Geometry and geometric operators are useful to study because, like the systems we hope to analyze, they are also linear operators. Insights from the simpler world of geometry then help us understand complex linear operators such as electrical and mechanical systems.

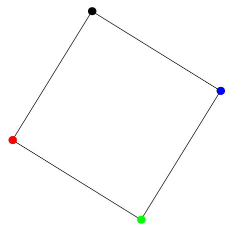
Analyze how the matrix works by looking at what it does to the unit square. Think of the unit square as three vectors: one from the origin to each corner. Here is what T does to each vector:



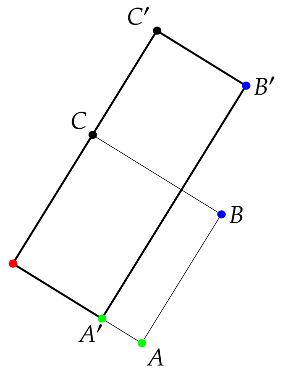
The square becomes a generic quadrilateral, and the vertex labels and colors show which corner in the square became which corner in the quadrilateral. What a messy shape! There must be an easier way to understand this transformation.

To find that understanding, divide and conquer. The square is composed of two basis vectors: $u = (1,0)$ and $v = (0,1)$. All four corners are a linear combination of these two vectors. But the transform T mixes u and v, which is why the output shape is funny. For example, Tu, which is the result of applying T to u, is $(1,1/2)$, which is along neither the x nor y axis. So the effect of T is complicated to describe because we are asking for its effect on u and v.

Perhaps there are vectors for which the effect of the transform T is easy to describe. Look at another unit square:



This specially chosen unit square is turned into a rectangle by the transform T:



The green dot (vertex A) moves inwards *along its line* and the black dot (vertex C) moves outwards along its line. In a coordinate system where this square represents the x and y axes, the transform T has a simple effect.

The basis vectors of this special square, whose basis vectors are perhaps rescaled but are not rotated, are the **eigenvectors** of T. In German, *eigen* means own in the sense of ownership, so these vectors are said to belong to the operator T. Call one of the vectors u' and the other v'. Then

$$Tu' = \lambda_1 u'$$

and

$$Tv' = \lambda_2 v',$$

where λ_1 and λ_2 are the scaling factors for each direction. The scaling factors are called the eigenvalues, which are said to belong to the operator T. For these directions, the transform T turns into simple multiplication.

And here is the amazing consequence. Any vector is a linear combination of u' and v', so to find out what T does to any vector q, first write that vector as a sum of u' and v':

$$q = au' + bv'.$$

Then apply T to both sides:

$$Tq = T(au') + T(bv') = aTu' + bTv'.$$

Since Tu' is easy – it's just $\lambda_1 u'$ – and similarly for Tv', we get

$$Tq = a\lambda_1 u' + b\lambda_2 v'.$$

So we have decomposed T into two parts, each of which we know how to handle (because they are each simple multiplication). And we can write any vector as a combination of these parts. So we can understand how T behaves by using this eigenvector decomposition.

13.2 Sketching the frequency response

The systems in the previous chapters are, like the operator T, also linear operators. The operator T, being two dimensional, had two eigenvalues and eigenvectors, and each eigenvector had two components. Our systems are infinite-dimensional operators, even the discrete-time systems. So they have an infinite number of eigenvalues and eigenvectors, and each eigenvector has infinite length. An infinite-length vector is just a function. So instead of eigenvectors, our more complex linear systems have eigenfunctions. But the idea is identical to the idea in the geometry example: The eigenfunctions of a system are the special functions that are preserved when fed through the system. Finding the special functions decomposes a system into simple pieces.

Our systems have an infinite number of eigenvalues. Collect them into a function, the eigenvalue function. Given an eigenfunction, it tells you its eigenvalue: i.e. how much the system multiplies that eigenfunction when fed in as an input signal. For our linear systems, the eigenfunctions are the exponential functions e^{pt} (for all time). How do you know? Because they are the eigenfunctions of each block-diagram element. So the system function H is the eigenvalue function, meaning that $H(p)$ is the eigenvalue of e^{pt}!

The frequency response is a specialization of the eigenvalue function where we ask only about oscillating eigenfunctions. Those functions are the set $e^{j\omega t}$; from them we can also build sines and cosines as well if we need. The eigenvalue of the eigenfunction $e^{j\omega t}$ is $H(j\omega)$, which is a complex number. It can be broken into a real and imaginary part or into a magnitude and phase. The usual choice is magnitude and phase because the magnitude tells you how much the system multiplied the oscillating signal, and the phase tells you by what angle the system delayed the signal.

So sketching the frequency response $H(j\omega)$ requires finding its magnitude and its phase. Here we sketch the magnitude portion – the *magnitude response* – in the high-Q limit, and leave you to sketch the phase and then to sketch both in the low-Q limit.

In the high-Q limit, the denominator of the system function is

$$s^2 + \frac{s}{\omega_0 Q} + \omega_0^2.$$

To keep the ω_0's at bay, use the dimensionless variable $\bar{s} \equiv s/\omega_0$ instead of s. Then the denominator is

$$\bar{s}^2 + \frac{\bar{s}}{Q} + 1.$$

So the denominator of $H(j\bar{\omega})$ is

$$1 - \bar{\omega}^2 + \frac{j\bar{\omega}}{Q},$$

where $\bar{\omega} \equiv \omega/\omega_0$. Thus we need to sketch the magnitude function

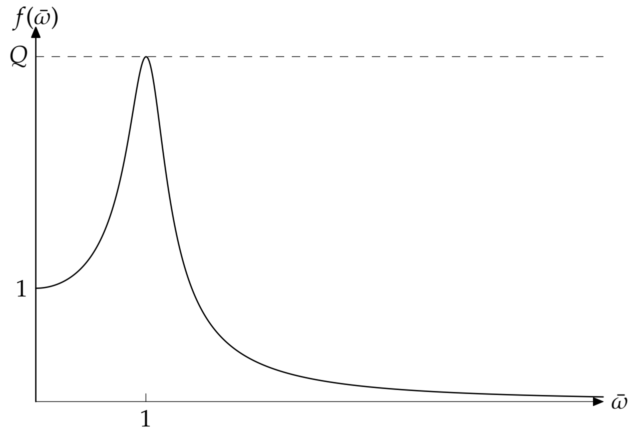
$$f(\bar{\omega}) = \frac{1}{|1 - \bar{\omega}^2 + j\bar{\omega}/Q|}.$$

In either the high-Q or the low-Q limit, the sketching does not require taking nasty square roots. Here we are doing the high-Q limit. Let's sketch it by looking at interesting values of $\bar{\omega}$. The first candidate is $\bar{\omega} = 0$, where $f(\bar{\omega}) = 1$. Let's get a slightly better estimate so that we know how f behaves in the region near 0. For $\bar{\omega} \approx 0$, the $j\bar{\omega}/Q$ term can be neglected in the high-Q limit. Then $f(\bar{\omega}) \approx 1 + \bar{\omega}^2$.

Exercise 67. How can we neglect a linear $\bar{\omega}$ term yet keep a quadratic $\bar{\omega}$ term?

The next $\bar{\omega}$ region is $\bar{\omega} \to \infty$, whereupon $f(\bar{\omega}) \to 0$. The most interesting $\bar{\omega}$, however, is the one that makes the real part of the denominator vanish, which is when $\bar{\omega} = 1$. Then the denominator becomes j/Q and $f(\bar{\omega}) = Q$. Since Q is huge, the gain is huge at this special frequency $\bar{\omega} = 1$, and it is probably the maximum because choosing $\bar{\omega} = 1$ wipes out a large part of the denominator.

To summarize: When $\bar{\omega} = 0$, the gain starts at 1 and rises quadratically around $\bar{\omega} = 0$. At $\bar{\omega} = 1$, the gain hits a probable maximum at Q, and then falls to zero as $\bar{\omega} \to \infty$. Here is a sketch that meets the requirements:



This frequency response is sharply peaked, so it is used to make systems that select only a limited range of frequencies – for example, an analog radio uses a resonant circuit to select one radio station from the many that occupy the radio spectrum. When you turn the frequency dial on the radio, you change ω_0 and (roughly) keep Q constant.

Exercise 68. Be more careful with the analysis near $\bar{\omega} = 1$ to find whether the maximum is slightly before, slightly after, or right at $\bar{\omega} = 1$ and to find the width of the peak.

Exercise 69. Sketch the phase response using the vector method from lecture. Use extreme cases of $\bar{\omega}$, then sketch a smooth interpolation.

Exercise 70. Sketch the magnitude response for $Q = 0.1$ (the low-Q limit).

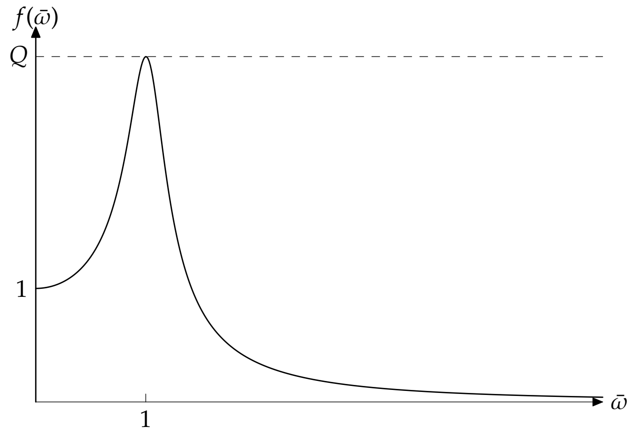
The $\bar{\omega}$ axis in these plots, it turns out, has a flaw. To see what it is, look at the magnitude sketch in the high-$\bar{\omega}$ limit. Because you used the analytic form, you know that it falls as $1/\bar{\omega}^2$. However, the sketch shows merely a generic decay to zero. For all you can tell from the sketch, the decay could come from $e^{-\bar{\omega}}$ or $\bar{\omega}^{-4}$ or a host of other possibilities. So we would like a method of sketching that makes explicit the limiting behavior and other useful information. That method is the purpose of Bode plots, the subject of the next chapter.

Recitation 14
Bode plots

After studying this chapter, you should be able to:

- sketch a Bode plot given a system function $H(s)$;
- deduce a system function from a Bode plot.

Here is the magnitude sketch of the low-pass, second-order, high-Q system from the last chapter, where $\bar{\omega}$ is the scaled frequency ω/ω_0:



In the high-$\bar{\omega}$ limit, the magnitude is proportional to $1/\bar{\omega}^2$. The sketch, however, shows merely a generic decay to zero. The decay could come from $e^{-\bar{\omega}}$ or $\bar{\omega}^{-4}$ or many other possibilities. We would like a method of sketching that makes explicit the limiting behavior and other useful information. Bode plots, the subject of this chapter, are that method.

Bode plots provide quick insight into a system's frequency response. Since straight lines are faster to sketch and analyze than are curves, Bode plots are constructed mostly from straight-line approximations to the exact magnitude and phase curves.

The first step in the approximation is to choose useful axes. For the magnitude sketch, the usual linear–linear axes hides behaviors such as $|A| \sim \bar{\omega}^{-2}$. However, log–log axes turn that behavior into a straight line. To see why, take the logarithm of both sides:

$$\ln|A| = -2\ln\bar{\omega} + \text{constant}.$$

In general, a log-log plot turns a $y = x^n$ dependence into a straight line with slope n. Therefore, for the magnitude sketch, a Bode plot uses log–log axes.

Next we decide the axes for the phase sketch. The phase θ and magnitude $|A|$ live together in the complex-valued gain $A = |A|e^{j\theta}$. Its logarithm is

$$\ln A = \ln|A| + j\theta.$$

The log-magnitude is the real part of $\ln A$, and the phase θ is the imaginary part of $\ln A$. So phase and log-magnitude are similar. This similarity suggests that if we are sketching log-magnitude for the magnitude graph, then we should sketch phase itself for the phase graph (rather than sketching log phase). As in the magnitude sketch, the frequency axis of the phase sketch will be logarithmic.

In summary, a Bode plot has two parts: a magnitude sketch on log–log axes and a phase sketch on linear–log axes. Since systems can be factored into poles and zeros, we derive Bode plots for isolated poles and zeros, making convenient straight-line approximations to the exact shapes. The extension to complex poles, such as in second-order systems with $Q > 0.5$, is a (challenging) exercise for the reader. The straight-line approximations that follow are due to R. D. Middlebrook of Caltech's electrical-engineering department, and it is a pleasure to acknowledge his work here.

14.1 Single-pole magnitude sketch

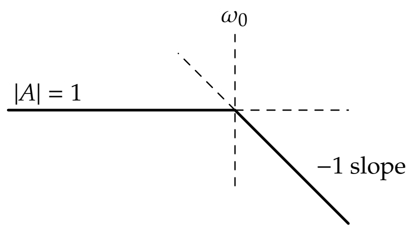
The factor for a single pole is

$$A(s) = \frac{1}{1+\frac{s}{\omega_0}},$$

where ω_0 is the corner frequency. For an RC circuit, for example, ω_0 is $1/RC$. The frequency response results from setting $s = j\omega$. The gain is then

$$A(j\omega) = \frac{1}{1+j\frac{\omega}{\omega_0}}.$$

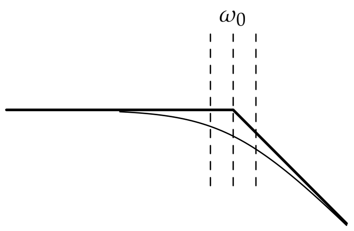
To sketch the magnitude, look at the extreme cases of ω. As $\omega \to 0$, the gain becomes 1. So in the low-ω limit, the magnitude sketch is a flat line. In the other extreme of $\omega \to \infty$, the 1 in the denominator is tiny compared to the other term $j\omega/\omega_0$. So the gain becomes $\omega_0/(j\omega)$ and its magnitude is ω_0/ω. On log–log axes, this dependence turns into a straight line with a -1 slope. At $\omega = \omega_0$ the two extreme cases agree (and are both somewhat inaccurate), so the two straight lines intersect at $\omega = \omega_0$. The magnitude sketch is therefore:



Let's compare it with the exact curve. The magnitude is

$$|A(j\omega)| = \left(1+\frac{\omega^2}{\omega_0^2}\right)^{-1/2}.$$

Here is a sketch using log–log axes that includes this curve:



Beyond a factor of 2 from the corner frequency (indicated by the outer vertical lines), the exact sketch hardly deviates from the straight-line sketch.

Exercise 71. What is the maximum error from using this straight-line approximation to the exact curve? Express your answer as a fraction and in dB.

14.2 Single-pole phase sketch

Now work out the corresponding phase sketch. Look at the phase of

$$A(j\omega) = \frac{1}{1 + j\frac{\omega}{\omega_0}}$$

in the extreme cases of ω. As $\omega \to 0$, the imaginary piece in the denominator vanishes, making the gain real and positive so the phase is 0. As $\omega \to \infty$, the 1 in the denominator is negligible compared to the $j\omega/\omega_0$ term, so the gain has a factor of $1/j$, which is a phase of $-90°$. In between, at $\omega = \omega_0$, the gain is $1/(1 + j)$, which is a phase of $-45°$. So:

$$\theta = \begin{cases} 0° & \text{as } \omega \to 0; \\ -45° & \text{when } \omega = \omega_0; \\ -90° & \text{as } \omega \to \infty. \end{cases}$$

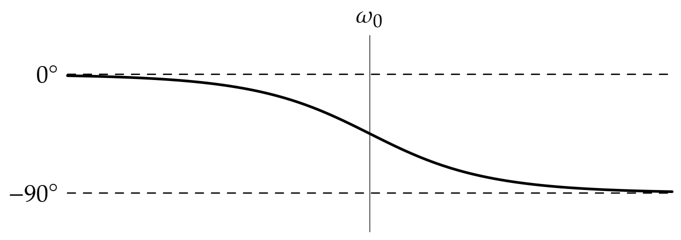
These values give the two horizontal asymptotes for θ and a point on the line connecting them (at $\omega = \omega_0$). To finish the sketch, we need to estimate the slope of that connecting line. To do so, look at the analytic expression for the phase:

$$\theta = -\tan^{-1}\frac{\omega}{\omega_0}.$$

This form is for a linear–linear plot because it uses frequency directly. To draw the curve on the linear–log axes, write the phase in terms of the logarithmic coordinate $l = \log(\omega/\omega_0)$. Then

$$\theta = -\tan^{-1} e^l.$$

It looks like



Its slope is

$$\frac{d\theta}{dl} = -\frac{e^l}{1+e^{2l}}.$$

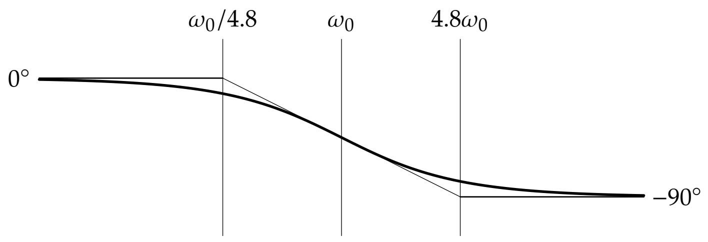
At $\omega = \omega_0$, the logarithmic coordinate is $l = 0$, so the slope is $-1/2$. That line would make the phase go from $-45°$ to $-90°$ in a logarithmic distance

$$\Delta l = \frac{\pi/4}{1/2} = \pi/2.$$

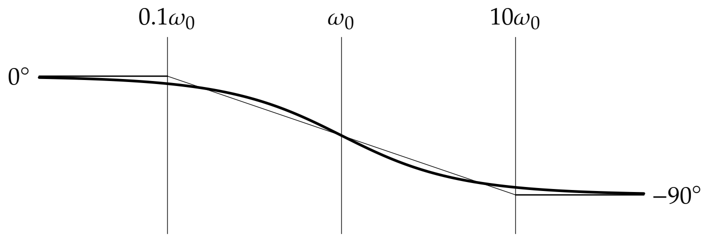
This logarithmic distance corresponds to a frequency ratio of $e^{\pi/2} \approx 4.8$.

Exercise 72. [very hard, perhaps unsolved!] Is $e^{\pi/2}$ a transcendental number?

The picture is then

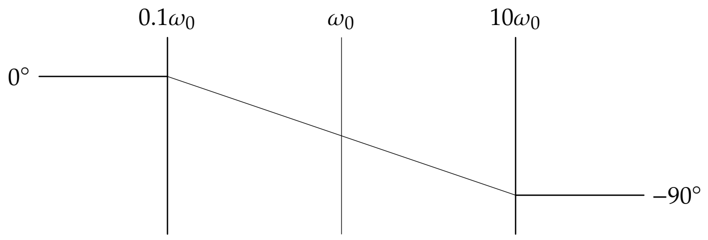


This straight-line approximation matches the slope of the actual phase at $\omega = \omega_0$. However, it pays for this accuracy by being inaccurate at the asymptote intersections. If the goal is to match the phase itself rather than to match its slope, it is worth sacrificing accuracy in the slope of the phase to get accuracy in the phase. To do that, make the connecting line less steep. A convenient choice is to make it go from $\omega_0/10$ to $10\omega_0$:



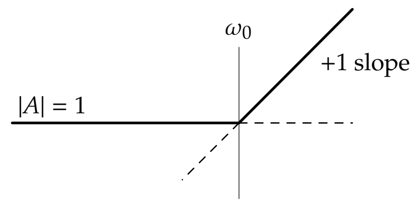
Exercise 73. What is the maximum error (in degrees) from using this piecewise-linear approximation to the exact phase?

This sketch leads to a useful design rule: An isolated pole affects the phase for a decade – a factor of 10 – on either side of the pole. The phase sketch with only the asymptotes is

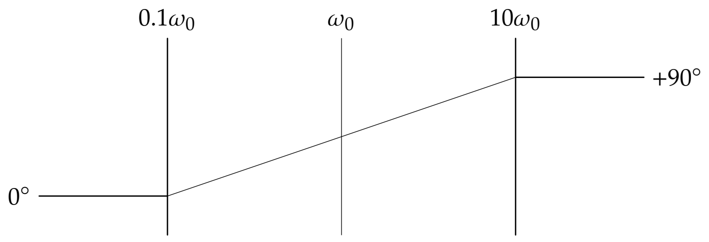


14.3 Zeros

A zero is the reciprocal of a pole. Logarithmic coordinates turn multiplication into addition and division into subtraction. So to sketch the magnitude due to a zero, negate the magnitude sketch due to a pole. Similarly, to make the phase sketch for a zero, negate the phase sketch for a pole. Here is the resulting magnitude:



Here is the resulting phase:

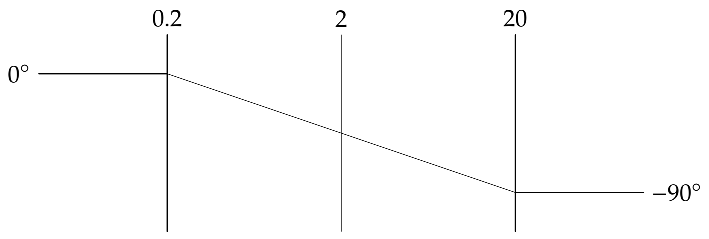


14.4 Combining sketches

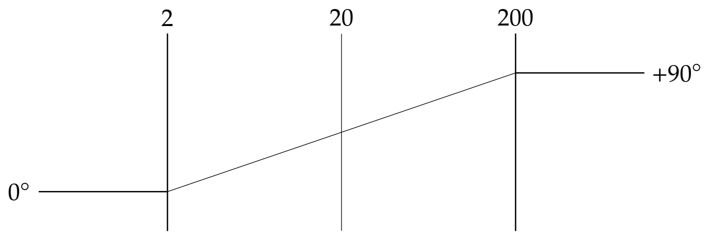
To illustrate how to combine sketches, we sketch the phase for the system

$$H(s) = \frac{1 + \frac{s}{20}}{1 + \frac{s}{2}}.$$

The pole contributes the negative-phase shape centered on $\omega = 2$:



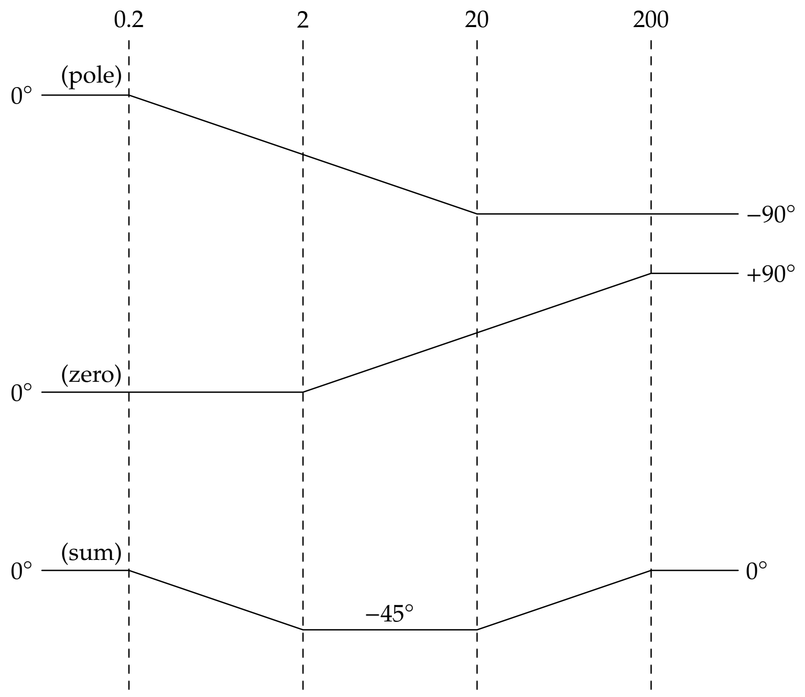
The zero contributes the positive-phase shape centered on $\omega = 20$:



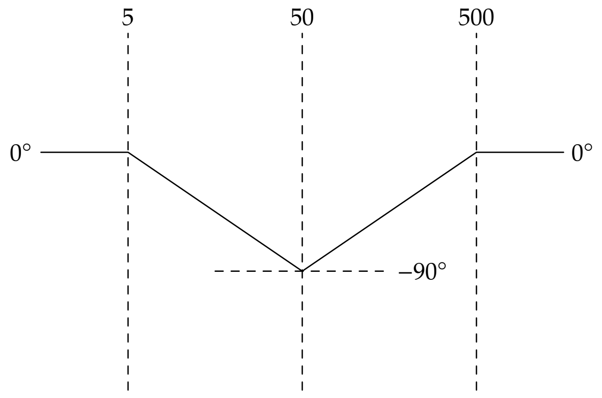
Align and add these sketches to get the phase for the whole system. Since each sketch is piecewise linear and continuous, their sum will also be piecewise linear and continuous. To construct the sum, find the sum at the corners of either sketch, then sketch the piecewise linear curve through them those points. The corners are at 0.2 (from the pole), 2 (from both), 20 (from both), and 200 (from the zero), and the sums are computed as follows:

ω	θ_{pole}	θ_{zero}	θ_{sum}
0.2	0°	0°	0°
2	−45	0	−45
20	−90	+45	−45
200	−90	+90	0

The continuous, piecewise-linear sketch through those points is the third graph in the stack below:



Exercise 74. Give a system function with this phase sketch:

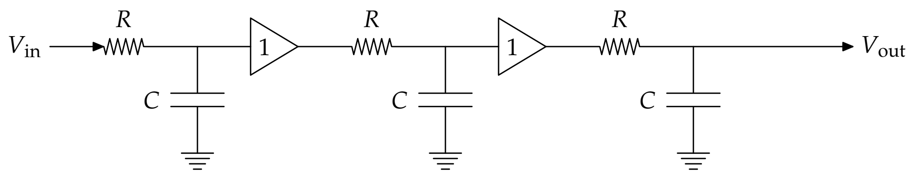


Recitation 15
Feedback and control

After studying this chapter, you should be able to:

- sketch Bode plots for complicated systems by making plots for simpler subsystems and combining these plots; and
- use Bode plots to analyze feedback control.

We illustrate feedback and control by improving the performance of this system:[1]



This triple-*RC* cascade, and how it behaves when wrapped in a feedback loop, is familiar from recent homework problems on the Hewlett–Packard oscillator. In those problems the goal was to use feedback control to turn the system into an oscillator. An ideal oscillator does not care about its input: It just oscillates at a predictable frequency and amplitude no matter what input it is given. In this chapter, we study the opposite problem: to make the system into a follower. A perfect follower copies its input to its output, whereas this system does so only for a limited range of input frequencies. In this chapter, you learn how to use feedback control to extend that range. This goal might seem pointless, since a wire is a simple and almost-perfect follower. But the input signal might live in one domain – for example the desired position of a robot arm – and the output signal can live in another domain – for example, the actual position of a robot arm. A perfect follower would be the ideal robot-arm controller.

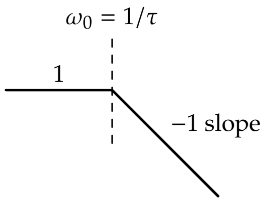
The system, before we add feedback, is three *RC* sections isolated from one another by unity-gain buffers, so the system function is the cube of the *RC* system function:

$$H(s) = H(s)^3_{\text{RC}} = \left(\frac{1}{1+\tau s}\right)^3$$

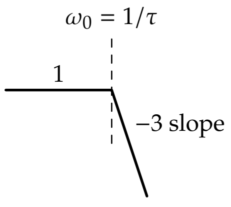
Pause to try 54. Sketch its magnitude on the usual Bode axes.

Here is the magnitude sketch for a $1/(1+\tau s)$ subsystem and for a cascade of three:

[1] Thanks to Jeff Lang for suggesting this problem and for the idea of using lead–lag networks to stabilize it.



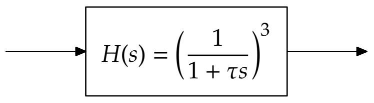
one RC section



three sections

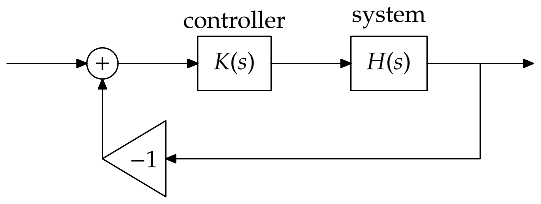
The sketch for the triple cascade is just three times the sketch for one RC section, thanks to the magical properties of the logarithm, which turns multiplication into addition and exponentiation into multiplication.

The bandwidth is the frequency range over which the system behaves like a follower. An ideal follower has unity gain, so our system acts like a follower until the corner at $\omega = \omega_0$ makes the gain fall. To extend this frequency range, we wrap the system in a negative-feedback loop. Since we will be studying many variants of this system, rather than redrawing all the resistors and capacitors, we *abstract* it into a single block represented by $H(s)$.



15.1 A possible solution using proportional feedback

A general setup for feedback control is this block diagram:



The problem is to choose the controller, represented by its system function $K(s)$, that makes the closed-loop system into a decent follower. Black's equation tells us that the closed-loop system function is

$$A(s) = \frac{\overbrace{K(s)H(s)}^{\textit{forward gain}}}{1 - \underbrace{(-K(s)H(s))}_{\textit{loop gain}}} = \frac{K(s)H(s)}{1 + K(s)H(s)}.$$

The simplest controller to analyze is a constant gain $K(s) = K_0$ which produces this closed-loop gain:

$$A(s) = \frac{K_0 H(s)}{1 + K_0 H(s)}.$$

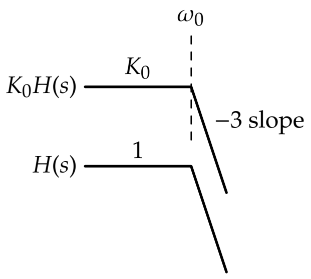
Look at the extreme cases of K_0. When K_0 is tiny, the denominator is approximately 1 so the closed-loop gain is roughly $K_0 H(s)$. This system function has two problems. First, the gain is tiny, even at low frequencies, whereas an ideal follower has unity gain. Even if we do not object to that problem, which we might be able to fix by using a post-amplifier, we also have not improved the bandwidth of the closed-loop system.

When K_0 is large, then the denominator is roughly $K_0H(s)$, which equals the numerator, so the closed-loop system is roughly $A(s) = 1$. Great! So use a decent op-amp as the controller and you are done.

Now we use Bode magnitude plots to find the bandwidth as a function of the gain K_0, assuming that $K_0 \gg 1$. We do so by sketching the magnitudes of the numerator and denominator of the closed-loop gain.

Pause to try 55. Sketch the Bode magnitude for $A(s)$ when $K(s) = K_0$.

The numerator of $A(s)$ is $N(s) = K_0H(s)$ so the numerator's plot is identical to the plot for $H(s)$ except that the gain of K_0 shifts the plot upward. The plots in the margin show how to get $K_0H(s)$ from $H(s)$.



Pause to try 56. Sketch the Bode magnitude for the denominator $D(s)$, again when $K(s) = K_0$ and $K_0 \gg 1$.

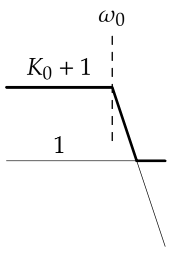
Sketching the magnitude of the denominator is tricky because the denominator $D(s)$ is the sum of two terms. Logarithms, and logarithmic plots, are ideal when multiplying terms – i.e., when cascading systems – because logarithms convert a product into a sum. However, what happens to the logarithm of a sum?

Fortunately you solved that problem in sketching the magnitude for a single RC section. Its frequency response is $H(j\omega) = 1/(1 + j\omega\tau)$ and it has magnitude $(1 + \tau^2\omega^2)^{-1/2}$. So its logarithm is

$$-\frac{1}{2}\log(1 + \tau^2\omega^2),$$

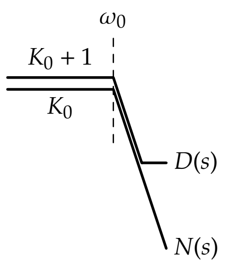
which contains the logarithm of a sum. We sketched it by looking at the extreme cases of ω. When $\tau\omega \gg 1$, the sum is dominated by $\tau\omega$. When $\tau\omega \ll 1$, the sum is dominated by 1. In both extremes, the sum disappears making it easy to take the logarithm.

Similarly, to sketch the magnitude of $D(s) = 1 + K_0H(s)$, look at the sum for extreme values of $K_0H(s)$. When $K_0H(s) \gg 1$, the 1 is negligible and the magnitude of the sum tracks the magnitude of $K_0H(s)$. When $K_0H(s) \ll 1$, the $K_0H(s)$ is negligible and the magnitude of the sum tracks the 1. So, the magnitude sketch for $D(s)$ tracks the larger term at that frequency. The figure in the margin shows this graphical construction. Its accuracy is slightly improved by using $K_0 + 1$ as the DC gain rather than just K_0.

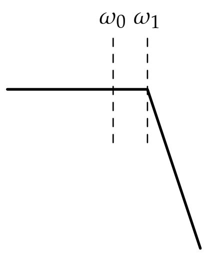


Pause to try 57. Sketch the magnitude of $A(s)$ by combining the sketches for $N(s)$ and $D(s)$.

Since $A(s) = N(s)/D(s)$, the magnitude sketch (using the log–log axes) for $A(s)$ is the difference between the sketches for $N(s)$ and $D(s)$. Here are those sketches stacked on top of each other and then their difference:



$N(s)$ and $D(s)$



$N(s)/D(s)$

To take the difference, sweep from left to right in frequency. At low frequency, the numerator is K_0 and the denominator is K_0+1, so the closed-loop gain is $K_0/(K_0+1) = 1/(1+1/K_0)$. That argument works until ω_0, when both sketches start falling and we have to think again about the difference. But both sketches fall with a -3 slope, so their difference does not change. Thus the gain remains $1/(1+1/K_0)$ until $D(s)$ stops falling. That change happens at a new corner frequency ω_1. To find it, find where the magnitude of $D(s)$ crossed 1. At ω_1, the magnitude of $D(s)$ starts falling from K_0+1 with a -3 slope. So it reaches 1 at $\omega_1 = \omega_0(K_0+1)^{1/3}$. The cube root arises because the horizontal distance (the frequency axis) is one-third of the vertical drop (the gain ratio K_0+1), and on a log scale the one-third reflects a cube root.

So, for large K_0, the gain is almost 1, which is good, and its bandwidth has grown by a factor of $(K_0+1)^{1/3}$ to become $\omega_0(K_0+1)^{1/3}$, which is also good.

15.2 Instability!

However, the bad news is that we have ignored stability. For discrete-time systems, we saw that feedback produces instability when the proportional gain becomes large. In discrete-time systems, instability means that the poles crossed outside the unit circle. In a continuous-time system, instability means that the poles of the system cross into the right half of the s plane. In the last homework, you found that this crossing happened when $K_0 = 8$.

Another route to the same result is the *Nyquist criterion*, which says that the closed-loop system is barely stable when the magnitude of the open-loop system crosses 1 at the same frequency at which its phase crosses $\pm 180°$. To make sense of this criterion, look at the closed-loop gain with an arbitrary controller:

$$A(s) = \frac{K(s)H(s)}{1+K(s)H(s)}.$$

When $A(s)$ is barely unstable, the system has a pole, or poles, on the imaginary s axis. The imaginary s axis is the ω axis, so the denominator $1+K(j\omega)H(j\omega)$ will be zero for some frequency ω. The open-loop gain is $K(j\omega)H(j\omega)$, so when the open-loop gain is -1 the system is barely unstable (or barely stable, depending on whether you are an optimistic or a pessimist). When $K(j\omega)H(j\omega)$ is -1, it has a magnitude of 1 and a phase of 180°, which explains the Nyquist criterion.

A more intuitive although perhaps less rigorous justification is to think about the feedback loop. As long as the feedback stays negative, the system will partly correct errors in the output; for example, it will prevent unbounded oscillations. The larger the gain of the feedback loop, the more completely the system corrects errors in the output. However, when the loop phase becomes $\pm 180°$, the loop gets an additional minus sign, which turns the negative feedback into positive feedback. This positive feedback is not fatal if the gain around the loop is less than 1 (in magnitude), because each trip around the loop contributes a smaller signal, and the the signals make a convergent geometric series. However, if the gain around the loop is greater than 1, then the sum diverges: The system has become unstable. The boundary between stability and instability is when the gain around the loop is exactly 1 in magnitude at the frequency at which the phase is $\pm 180°$.

Pause to try 58. Apply the Nyquist criterion to our system to show that it goes unstable when $K_0 > 8$.

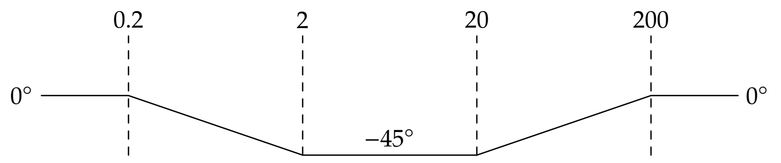
Let's apply the Nyquist criterion to our system to confirm the pole–zero analysis. Here the open-loop system is the controller followed by the triple RC, and its system function is $K_0/(1+\tau s)^3$. Its phase goes to $-180°$ when each RC contributes $-60°$ of phase. The phase from an RC circuit is $-\tan^{-1}(\omega/\omega_0)$, so the frequency for $-60°$ phase is $\sqrt{3}\omega_0$. At that frequency, each RC contributes a magnitude $(1+\omega/\omega_0)^{-1/2}$, which is $1/2$ when $\omega/\omega_0 = \sqrt{3}$. So when $K_0 = 8$, the open-loop gain of the controller followed by three RC sections is 1 at the $-180°$-phase frequency. Therefore the feedback system is barely stable when $K_0 = 8$, confirming the result from the pole–zero analysis in the homework.

The bandwidth of the controlled system with feedback is proportional to the cube root of $1 + K_0$, so the bandwidth cannot be greatly increased without making the system unstable.

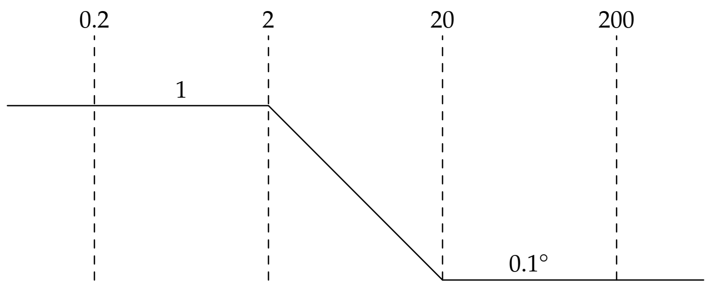
15.3 Improving the bandwidth

One way to improve the bandwidth is to use a more subtle controller. If the controller contributed positive phase without contributing gain, then it could raise the phase at the frequency $\sqrt{3}\omega_0$ and allow us to increase ω. A circuit cannot contribute positive phase at all frequencies, let alone do so without contributing gain because such a circuit would be a time machine. But if it can do its magic in an important frequency range, then it might suffice for our purposes.

In the last chapter, we sketched the magnitude and phase plots for a so-called lag–lead network: $(1 + \frac{s}{20}/(1+\frac{s}{2})$. It had this phase sketch:



Its magnitude sketch is not hard to make. It is flat until the pole at $\omega = 2$, then it starts falling until the zero at $\omega = 20$ raises the magnitude back to a flat line:



The reciprocal of the lag–lead network is a lead–lag network, and it has unity gain between $\omega =$ 0.2 and 2 and contributes positive phase in that range – just the behavior that we want! We need only slide the lead–lag network to the correct frequency to help stabilize our system. As one stage in the controller we therefore use this system:

$$B(s) = \frac{1 + s/\omega_2}{1 + s/(10\omega_2)},$$

and choose ω_2. The idea is that we add a lead–lag network (or lead–lag networks) to the controller, which still has the gain K_0, and enabling us to raise K_0 while keeping the system stable.

We leave it as an exercise for the reader to use Bode plots to answer the following:

Exercise 75. What ω_2 would you choose? Would you use more than one lead–lag network (assuming that all are identical)?

Recitation 16
Multiple representations of resonance

After studying this chapter, you should be able to:

- Find Q given a system function, and write a system function with a given Q;
- Find Q given the poles, and find the poles given Q;
- Find Q from how amplitude decays, and sketch an amplitude decay for a given Q;
- Find Q from a Bode magnitude sketch, and sketch the magnitude for a given Q; and
- Find Q from a Bode phase sketch, and sketch the phase for a given Q.

In short, the goal is to give meaning to Q by using pictures.

The quality factor Q, which measures how resonant a system is, is the important parameter for describing a second-order system. The other two parameters of the system are scale parameters that do not qualitatively change how the system behaves (see **Section 12.4** for the details). Because Q is so important, this chapter discusses four pictures for Q in resonant (high-Q) systems.

16.1 System function

The pictures can all be derived from the system function. Here is a generic low-pass, second-order system function:

$$H(s) = \frac{A_0}{1 + as + bs^2},$$

where A_0 is the zero-frequency gain, and a and b are coefficients that depend on the system. In an LRC circuit, a often contains the resistance and capacitance, and b often contains the capacitance and inductance. In a damped spring–mass system, a typically contains the damping constant and the mass, and b typically contains the spring constant and the mass.

But this form hides Q. Let's rewrite the denominator in steps. First define the corner frequency ω_0 as $1/\sqrt{b}$ so that the first and last terms of the denominator are 1 and $(s/\omega_0)^2$. The ratio s/ω_0 is dimensionless, which suggests defining a dimensionless s as $\bar{s} \equiv s/\omega_0$. Then the denominator becomes

$$1 + a\omega_0\bar{s} + \bar{s}^2.$$

The combination $a\omega_0$ is dimensionless. Since the linear-s term arises from the damping – for example, from the resistor in an LRC circuit – the dimensionless coefficient $a\omega_0$ measures how strong the damping is. Engineers traditionally use its reciprocal and measure how weak the damping is. Its reciprocal is also dimensionless and is the quality factor Q:

$$Q \equiv \frac{1}{a\omega_0}.$$

Then the denominator is

$$1 + \frac{\bar{s}}{Q} + \bar{s}^2.$$

It has only one parameter! The other two parameters of the system are scale parameters:

1. ω_0, the corner frequency, which is now hidden in $\bar{s}$;
2. A_0, the zero-frequency gain, which is in the numerator of $H(s)$.

Just as we hid the ω_0 scale parameter by rescaling the independent variable to be $\bar{s}$, we'll hide the scale parameter A_0 by rescaling the system function $H(s)$ to be $H(s)/A_0$. We will then study four pictures for the dimensionless system

$$\overline{H}(\bar{s}) = \frac{1}{1 + \frac{\bar{s}}{Q} + \bar{s}^2}.$$

In each picture we show where Q lives.

16.2 Poles

To find the poles, rewrite the denominator in factored form:

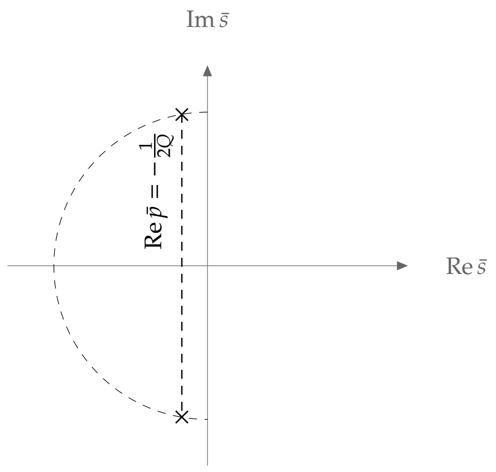
$$1 + \frac{\bar{s}}{Q} + \bar{s}^2 = (1 - \bar{s}/\bar{p}_1)(1 - \bar{s}/\bar{p}_2),$$

where $\bar{p}_1$ and $\bar{p}_2$ are the poles in the $\bar{s}$ plane. For high Q, the $\bar{s}/Q$ term is small, so the poles are those of $1+\bar{s}^2$, which are at $\pm j$. So the poles with a finite but large Q are near $\pm j$, which means that they are complex. Complex poles come in conjugate pairs, and they have identical magnitude. Since $\bar{p}_1\bar{p}_2 = 1$ (the coefficient of $\bar{s}^2$), it is also true that $|\bar{p}_1||\bar{p}_2| = 1$. The only way that the two poles can also have equal magnitude is if $|\bar{p}_{1,2}| = 1$. So the poles live on the unit circle in the $\bar{s}$ plane.

Because the $\bar{s}$ coefficient is $1/Q$, the sum of the poles is $-1/Q$. Since the poles are complex conjugates, their real parts are identical, so each has real part $-1/2Q$. The poles are therefore at

$$\bar{p} \approx -\frac{1}{2Q} \pm j.$$

Here is a picture of the poles for $Q = 3$:



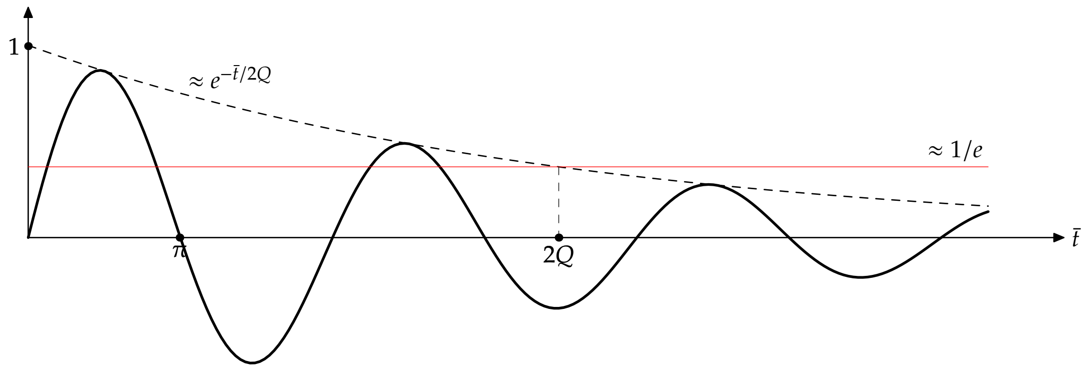
Exercise 76. Estimate Q from the pole–zero diagram.

16.3 Amplitude decay

The second picture for Q is the response of a high-Q system to an impulse. The negative real part in the poles means that disturbances (transients) eventually decay. The real part is tiny, so the decay takes a long time.

To make the sketch, use scaled axes. To get to the nondimensional unit system, multiply time by ω_0 to convert time into radians. The horizontal axis of the nondimensional sketch will be $\bar{t} = \omega_0 t$. The vertical axis will be $\bar{y}(\bar{t})$, where the scaled output $\bar{y}$ is defined as $\bar{y} \equiv y/A_0$. Since the scaled system function $\overline{H}(s)$ is dimensionless, $\bar{y}$ has the same dimensions as the input signal. An impulse $\delta(\bar{t})$ is dimensionless, so the output signal is also dimensionless. Thus the vertical axis will be dimensionless.

The negative real part of the pole contributes a $e^{-\bar{t}/2Q}$ factor to the impulse response, so transients decay with a (scaled) time constant of $2Q$. The $\pm j$ contributes an oscillation with unit frequency that lives inside the envelope (the dashed line) set by the exponential decay. When the decay has contributed a factor of $1/e$, the $\bar{t}$ axis has advanced $2Q$ radians:



Exercise 77. Estimate Q from the impulse response.

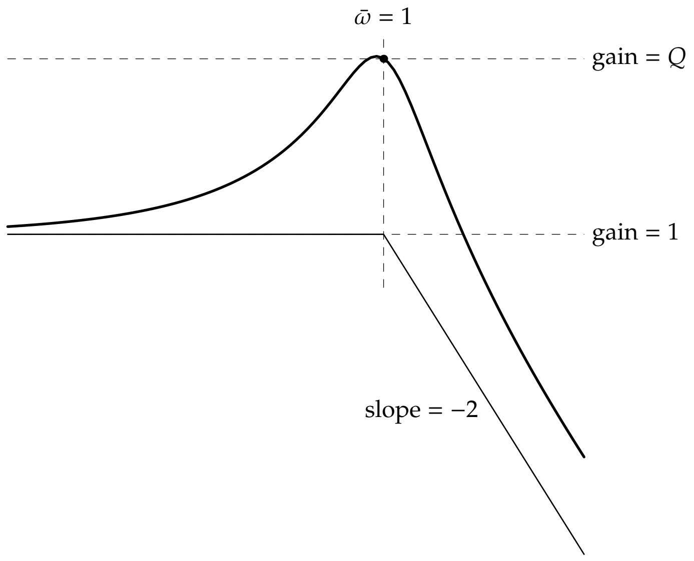
16.4 Magnitude plot

To sketch the magnitude, look at the extremes of $\bar{\omega}$. For low frequencies, which means $\bar{\omega} \ll 1$, the denominator is 1, so the magnitude is a flat line at 1. For high frequencies, which means $\bar{\omega} \gg 1$, the 1 and $\bar{\omega}/Q$ terms are small compared to $\bar{\omega}^2$, so the denominator is approximately $\bar{\omega}^2$. The magnitude is therefore $\bar{\omega}^{-2}$. On a Bode plot, the magnitude falls with a -2 slope. These extreme-cases calculations give the straight-line asymptotes.

The smooth curve follows the asymptotes far away from the resonance, i.e. far away from $\bar{\omega} = 1$. Near the resonance, the magnitude is large. The vector method for finding frequency response shows that the response has a large peak near $\bar{\omega} = 1$. At $\bar{\omega} = 1$, the magnitude is

$$|\overline{H}(j\bar{\omega})|_{\bar{\omega}=1} = \frac{1}{|1 - j/Q - 1|} = Q.$$

Although the asymptotes intersect at $\bar{\omega} = 1$ with a magnitude of 1, the smooth curve has a peak magnitude of approximately Q. Here is a Bode plot showing these features, using the usual logarithmic scales for $\bar{\omega}$ and magnitude:



16.5 Phase plot

The final picture for Q is in the phase. The phase of $\overline{H}(j\bar{\omega})$ is

$$\theta = -\tan^{-1}\left(\frac{\text{imaginary part of denominator}}{\text{real part of denominator}}\right),$$

where the minus sign in front of the arctangent reflects that the overall phase is the negative of the phase of the denominator. Put in the real and imaginary parts:

$$\theta = -\tan^{-1}\left(\frac{\bar{\omega}/Q}{1-\bar{\omega}^2}\right).$$

For low $\bar{\omega}$, the argument of the arctangent is roughly zero but is slightly positive, so the phase is roughly zero. For high $\bar{\omega}$, the argument of the arctangent is roughly zero but is slightly negative, so the arctangent is roughly 180° and the phase is roughly −180°. So the phase has two flat asymptotes, one at high frequency and one at low frequency.

Between those extremes, the phase shifts rapidly from one asymptote to the other. To find a straight-line approximation for that shift, first find the phase on Bode phase axes, which are logarithmic for frequency and linear for phase. Therefore define $\bar{\omega} = e^l$, where the l stands for 'logarithm'. Then

$$\theta = -\tan^{-1}\left(\frac{e^l/Q}{1-e^{2l}}\right).$$

The argument of the arctangent is

$$\frac{e^l/Q}{1-e^{2l}} = \frac{1}{Q}\frac{1}{e^{-l}-e^l} = -\frac{1}{2Q}\frac{1}{\sinh l}.$$

So

$$\theta(l) = \tan^{-1}\frac{1}{2Q\sinh l} = \cot^{-1}(2Q\sinh l).$$

A reasonable connecting asymptote is the line through the phase at $\bar{\omega} = 1$, which is $l = 0$, and with the same slope as the $\theta(l)$ curve. The slope at $l = 0$ is not hard to find. First approximate $\sinh l$ as l for small l. Then find the slope of $\cot^{-1} x$ for small x. For small x,

$$\cot^{-1} x \approx \frac{\pi}{2} - x,$$

as you can verify by taking the cotangent of both sides. At the origin, the slope of $\cot^{-1}$ is then -1, and therefore the slope of $\cot^{-1}(2Ql)$ at $l = 0$ is $-2Q$.

Thus the phase falls by π in a logarithmic distance

$$\Delta l = \frac{\pi}{2Q}.$$

That distance corresponds to a frequency ratio

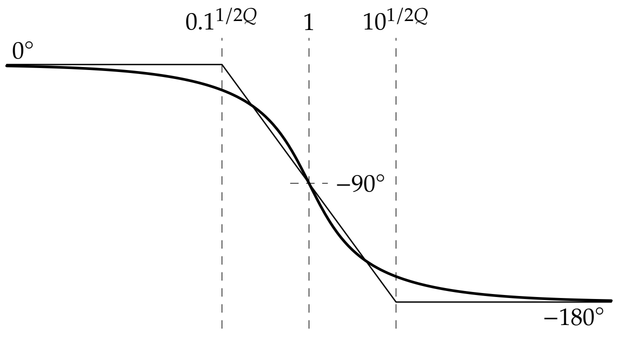
$$e^{\Delta l} = \left(e^{\pi/2}\right)^{1/Q}.$$

The factor $e^{\pi/2}$ is roughly 4.8. But we will not use that value directly. Instead we improve the asymptote and then interpret the result in words.

For a single pole, **Section 14.2** computed the phase asymptotes. The straight-line asymptote that matched the slope was too steep. It matched the slope of the phase at the cost of not matching the phase itself well. So we made it less steep by pretending that the slope was such that the phase fell $\pi/4$ over a decade change in frequency, instead of over a factor of 4.8 change in frequency.

Make the same approximation for the resonant system. Instead of saying that its phase changes by π in a frequency ratio of $4.8^{1/Q}$, say that it changes by π in a frequency ratio of $10^{1/Q}$. This approximation matches the single-pole result when $Q = 0.5$. For that Q, the second-order system splits into two identical first-order systems. At $\bar{\omega} = 1$, each first-order system contributes roughly $\pi/4$ phase lag per decade change in frequency. So two first-order systems contribute $\pi/2$ phase lag per decade change in frequency. The proposed rule for the resonant system says that its phase would fall by π over a frequency ratio of $10^{1/Q} = 100$ or equivalently by $\pi/2$ over a decade. This result agrees with the result from decomposing the second-order system into two single-pole systems.

Here is a sketch showing smooth phase curve and the three straight-line asymptotes: for low $\bar{\omega}$, for high $\bar{\omega}$, and for the transition between the extremes.

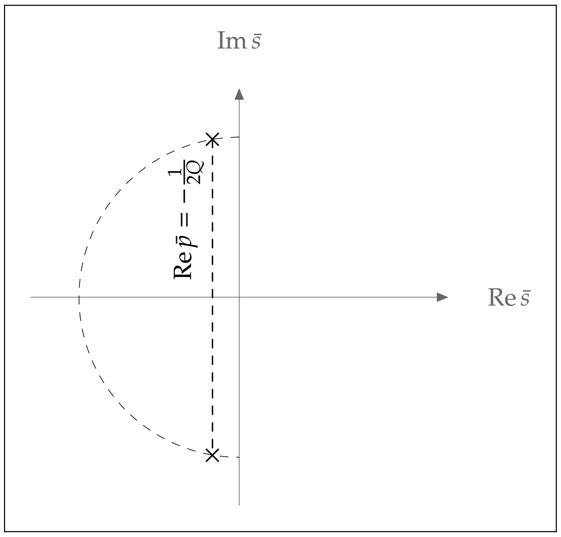


16.6 Summary

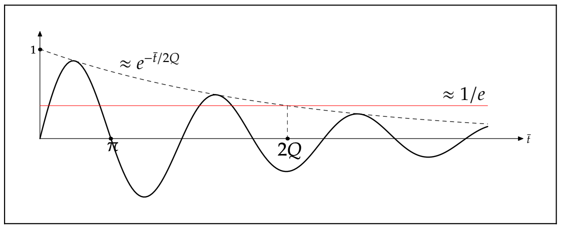
The canonical second-order system is

$$\overline{H}(\bar{s}) = \frac{1}{1 + \frac{\bar{s}}{Q} + \bar{s}^2}.$$

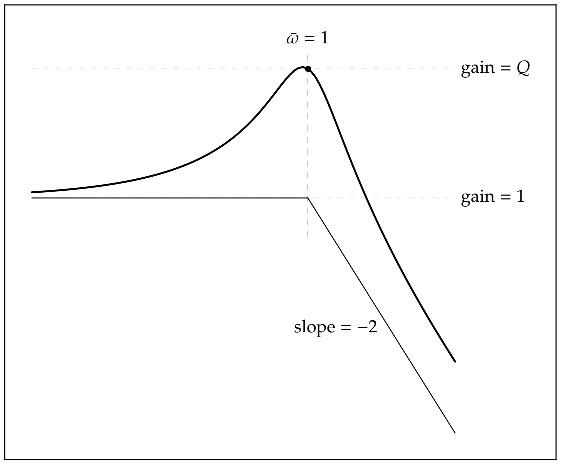
where $\bar{s} = s/\omega_0$ and $\overline{H} = H/A_0$. This system function is the symbolic face of Q. The next figures are the four pictorial faces of Q:



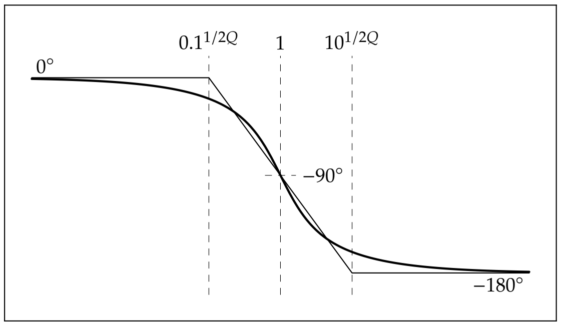
poles



amplitude decay



magnitude



phase

Recitation 17
Convolution

After studying this chapter, you should be able to:

- switch between the functional for a system (or the system function) and the signal representation;
- sketch the convolution of two signals; and
- compute the convolution of two signals analytically.

The theme of this course is *multiple representations*. Every fortnight we throw another representation at you. The newest representation for a system is by its impulse response. This representation, like any representation, makes some operations easier and others harder. It was easy to compose systems when representing a system by its functional or system function: Simply multiply the functions or functionals. In the impulse-response representation, the operation of *convolution* performs the same task. Convolution is not as simple as multiplication is, so in this chapter we develop intuitions for how it behaves.

But first a word about notation. The step function $u(t)$ often clutters integrals and other formulas, making it hard for me to think. Therefore, most formulas will omit the $u(t)$ if the context makes the meaning clear without $u(t)$. For example, the chapter might say that the impulse response of an RC circuit or leaky tank with $\tau = 1$ is e^{-t}, when the full story is that the impulse response is $e^{-t}u(t)$.

17.1 Convolution is a weighted average

The fundamental intuition about convolution is that the convolution $f \star g$ computes a weighted average of f where the weights are given by g. As an extreme example, take g to be the impulse $\delta(t)$, and f to be the impulse response of a leaky tank in units where the time constant $\tau = 1$, so $f(t) = e^{-t}$ for $t \geq 0$.

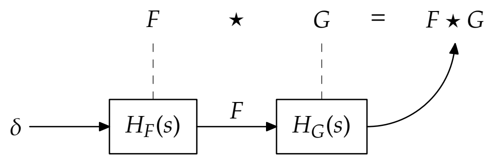
Pause to try 59. What is $f \star g$?

The weight function g is an impulse, which has all its weight at 0. Therefore the weighted average of f is just a copy:

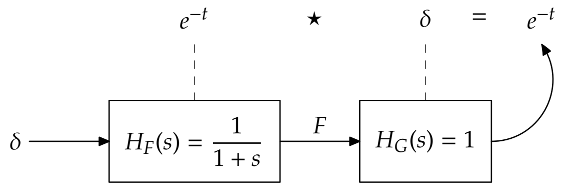
$$e^{-t} \star \delta(t) = e^{-t}.$$

That intuitive argument leads to a general result: Convolving any function f with an impulse reproduces f.

Let's check the specific result in two ways. The first way is from the origin of convolution as a way to compose systems. In the next figure, the dashed lines indicate that the signal F is a representation for the system with system function $H_F(s)$.



The signal F is the impulse response of a leaky tank with system function $1/(1+s)$. The signal $\delta(t)$ is the impulse response of a wire, which has system function 1. So the previous figure becomes



The cascade has system function

$$\frac{1}{1+s} \times 1 = \frac{1}{1+s}.$$

Its impulse response is e^{-t} so

$$e^{-t} \star \delta(t) = e^{-t}.$$

This systems method generalizes to the same conclusion: *Convolving any function f with an impulse reproduces f.*

The second way to check the intuitive result is by doing integrals. The principle of extreme laziness recommends avoiding integrals until you have tried every other method, wherefore we integrate only now. The convolution integral is

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau.$$

With $g(t) = \delta(t)$, the integral is

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(\tau) \delta(t-\tau)\, d\tau.$$

The delta function is nonzero only when $t = \tau$, so the delta function picks out $f(t)$:

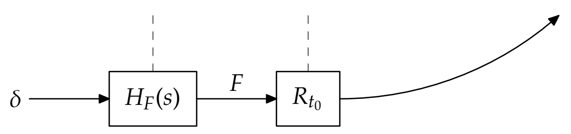
$$\int_{-\infty}^{\infty} f(\tau) \delta(t-\tau)\, d\tau = f(t).$$

This result, that convolving with a delta function does nothing, is the general result that we stated but did not quite prove by the averaging argument or by the systems method.

17.2 Convolving with a delayed impulse

Let's now try the next-simplest convolution example: Convolve $f(t)$ with the shifted impulse $\delta(t - t_0)$. The averaging method suggests that the result will be like f, perhaps with a shift. But the method does not make it obvious which direction to shift. So let's try the systems method. A shifted impulse corresponds to a wire with a delay, which is what the $\mathcal{R}$ operator does. In the discrete-time chapters, the $\mathcal{R}$ operator delayed a signal by the time step T. To make the amount of delay explicit, we subscript the $\mathcal{R}$ operator with the delay. Then $g(t) = \delta(t - t_0)$ is represented by the system $\mathcal{R}_{t_0}$. The convolution $f \star g$ is then $\mathcal{R}_{t_0} f(t) = f(t - t_0)$. This argument is represented by the following systems analysis:

$$f(t) \quad \star \quad \delta(t-t_0) \quad = \quad R_{t_0} f(t) \quad = \quad f(t-t_0)$$



So convolving f with the delayed impulse, which has all its weight at one point, reproduces f but with a delay.

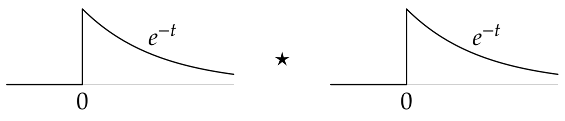
Check this result by doing the convolution integral. The integral is

$$(f \star g)(t) = \int f(\tau)\, \underbrace{\delta(t - \tau - t_0)}_{g(t-\tau)}\, d\tau,$$

which is $f(t - t_0)$ because the delta function picks out the $f(\tau)$ where $\tau = t - t_0$.

17.3 Two leaky tanks

Next convolve e^{-t} with a weight function that is more interesting than $\delta(t)$ or, if it is possible, is more interesting than $\delta(t - t_0)$. In particular, we convolve e^{-t} with itself and compute $e^{-t} \star e^{-t}$.



17.3.1 Qualitative analysis

Follow Wheeler's principle: Figure out the characteristics of the convolution before doing extensive calculations. The fundamental intuition is that convolution is a weighted average. In the extreme case of averaging f using an infinitely thin function (a delta function), f comes through unscathed. Convolving f with an exponential decay, on the other hand, will smooth f. Since f has a discontinuity, the smoothed version will probably have a discontinuity in its slope but itself be continuous. So at $t = 0$ the convolution should rise linearly from zero. Whereas for large t the convolution should decay to zero: Since f itself decays to zero, a smoothed version of f should do the same.

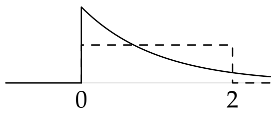
The next intuition is about the delay, or time shift of the convolution. Our averaging function, an exponential decay, begins at $t = 0$ and extends to the right. Extent can reasonably be defined as how long it takes the function to fall significantly, and e is a convenient choice for the significant factor. By this definition, g extends one time unit to the right, so it has some features of the delayed impulse $\delta(t - 1)$. The signal g is of course smoother than the delayed impulse, but when qualitatively working out the delay in $f \star g$, a delayed impulse is a useful approximation to g.

Then since $f(t)$ peaks at $t = 0$, convolving f with g will shift the peak right by one time unit. So $f \star g$ should peak when $t \approx 1$.

A related intuition is about the width and height of $f \star g$. Its width and height depend on the width and height of f and g. We would like approximations for f and g that have easily tunable height and width and that are easy to reason with. A useful approximation is a pulse because we can qualitatively predict the result of convolving pulses. To choose the width and height of the pulse, notice that convolution multiplies areas:

Exercise 78. Show that

$$\text{Area of } f \star g = \text{Area of } f \times \text{Area of } g.$$



So we should approximate f (or g) using a pulse that has the same area as f. Since f has unit area, the pulse's width should be the reciprocal of its height. The height should be a typical or average height, where the average is computed by weighting the actual height more strongly where f is more important (is larger). In this definition, the average height of f will be less than the maximum height of f. A reasonable guess is that the typical height is 1/2. So the width will be 2, producing the pulse approximation marked by the dotted line.

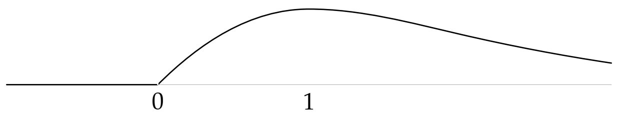
Pause to try 60. Sketch the convolution of the pulse with itself.

Convolving this pulse with itself produces a triangle. The triangle has a width (a base) of 4. You can see that result in two ways. You can do the integration, which was done in lecture. Or you can think qualitatively about convolution as averaging. Averaging f using g smears f by the width of g. This smearing should produce a function whose width is the sum of the widths of f and g. So the width of $f \star g$ should be 4. With a width of 4 and an area of 1, the triangle should have a height of 1/2.

We have reasoned our way to several qualitative conclusions about $e^{-t} \star e^{-t}$. The convolution should

1. rise linearly from zero at $t = 0$,
2. peak near $t = 1$,
3. decay to zero for large t, and
4. have a base width of roughly 4 and peak height of roughly 1/2.

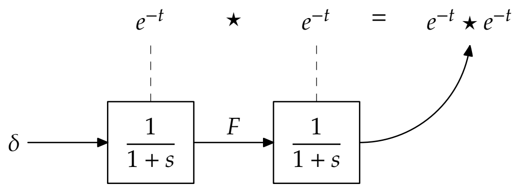
Here is a sketch that is consistent with these conclusions:



Next check the conclusions by finding the exact convolution.

17.3.2 Systems analysis

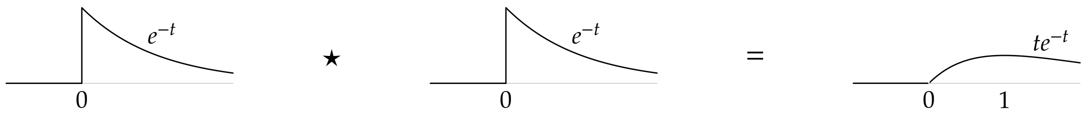
One way to find the exact convolution is a systems analysis:



The cascade has the system function $1/(1+s)^2$, which is a double pole at -1. Its impulse response is te^{-t}. So

$$e^{-t} \star e^{-t} = te^{-t}.$$

In pictures:



Let's check the qualitative conclusions against the expression te^{-t} and its picture. The function rises linearly from zero at $t = 0$, as predicted. It decays to zero for large t, also as predicted. It peaks at $t = 1$, as you can show by maximizing te^{-t} using differentiation. So the peak's location is also as predicted. It has a peak height of $1/e$, so the qualitative estimate of $1/2$ for the peak height is reasonably accurate, much more than we have a right to expect given the number of approximations that we made!

Exercise 79. Use systems analysis to show that $f \star g = g \star f$ for any f and g.

17.3.3 Integration

As a last resort, compute the convolution by integration. The integral is

$$\int f(\tau)g(t-\tau)\,d\tau.$$

Rather than strew the integral with lots of step functions (the horrible $u(t)$ notation), let's just figure out the correct integration limits. The integrand is nonzero only when $f(\tau)$ and $g(t-\tau)$ are nonzero. Our signals are nonzero only for positive time. So the integrand contributes something only when $\tau > 0$ (to make $f(\tau)$ nonzero) and $t > \tau$ (to make $g(t-\tau)$ nonzero). The two conditions restrict τ to the range $(0, t)$, and make the integral

$$\int_0^t e^{-\tau} e^{\tau-t}\,d\tau.$$

The e^{-t} factor that is part of $e^{\tau-t}$ is a constant when doing a $d\tau$ integral, so it can be pulled out. The remaining integrand is 1 integrated over a range of length t, so the convolution is

$$f \star g = \int_0^t e^{-\tau} e^{\tau - t}\, d\tau = te^{-t},$$

which agrees with the result from systems analysis.

17.4 Summary

The main message of this chapter is that you can sketch convolutions qualitatively before doing integrals. To make those sketches, several qualitative arguments are useful:

1. Convolution is a weighted average, and convolving (with a positive function) does a smoothing.
2. The width of $f \star g$ is the sum of the widths of f and g.
3. A delay in the averaging function (the g) delays the convolution $f \star g$.
4. Convolution multiplies areas:

$$\text{Area of } f \star g = \text{Area of } f \times \text{Area of } g.$$

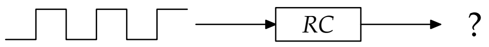
These qualitative arguments are particularly useful for understanding and designing complicated systems – for example, to understand the atmosphere and Hubble telescope lens together smear the image of the sky.

Recitation 18
Fourier series and filtering

After studying this chapter, you should be able to:

- explain the analogy between Fourier coefficients and coordinates in a vector space;
- find the Fourier coefficients of a periodic signal; and
- analyze a system by how it alters Fourier coefficients.

We illustrate Fourier series and filters using a square-wave input signal fed into an *RC* circuit:



The problem is to find the output signal. It can be found numerically and perhaps even analytically by solving the leaky-tank or *RC* differential equation. Because the output signal is knowable without using Fourier series or filtering, feeding a square wave to an *RC* circuit is useful for practicing and understanding Fourier series and filtering.

In this chapter, we first find the Fourier representation of the square wave. Then we can compute the output signal in two ways – using the time representation or the Fourier representation – and can compare the results. Rather than exactly solving differential equations or calculating general and therefore messy Fourier sums, we'll analyze the system in the two extreme cases of a fast and a slow square wave.

18.1 Fourier representation of signals

In the Fourier representation, the square wave is written as a weighted sum of oscillating exponentials (or of sines and cosines). In this representation, the circuit *filters* the signal: It adjusts the weights of the exponentials to produce the Fourier representation of the output signal. A system's operation as a filter is often simpler to understand than its operation as a differential equation.

The Fourier representation is

$$\text{square wave} = \sum_{k=-\infty}^{\infty} a_k F_k$$

where a_k is the k^{th} Fourier coefficient or weight; F_k is the k^{th} Fourier basis function, given by

$$F_k \equiv e^{j\omega_k t};$$

and $\omega_k \equiv 2\pi k/T$ is the angular frequency of the k^{th} Fourier function. The Fourier weights a_k that represent a function $f(t)$ are given by the Fourier inversion formula:

$$a_k = \frac{1}{T}\int_T f(t)e^{-j\omega_k t}\,dt,$$

where T is the period of the function $f(t)$ and, as a subscript on the integral, indicates integration over one period.

18.1.1 Function space

To understand these formulas, think of the Fourier representation as a coordinate system in *function* space, which is a kind of vector space. In a finite-dimensional vector space, for example the Euclidean plane, any vector r is a weighed sum of the unit vectors $\hat{x}$ and $\hat{y}$ along the coordinate axes. In that form,

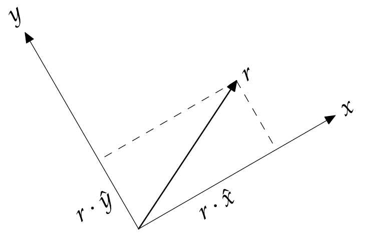
$$r = a\hat{x} + b\hat{y},$$

where a and b are the coordinates or weights. In function space, the coordinate axes of this space are the functions F_k, and the coordinates of $f(t)$ are its Fourier weights a_k. Then the Fourier-representation formula

$$f(t) = \sum_{k=-\infty}^{\infty} a_k F_k$$

is an infinite-dimensional version of the familiar two-dimensional form.

How do you compute the weights a_k? In a finite-dimensional vector space with perpendicular axes, you find each coordinate of a vector by computing its dot product with a unit vector along that axis. Similarly, to find the coordinates of a function $f(t)$ in this function space, take the dot product of $f(t)$ with each basis vector F_k. The dot product of two vectors $b = (b_0, b_1, b_2, \ldots)$ and $c = (c_0, c_1, c_2, \ldots)$ is the sum of componentwise products:



$$b \cdot c = \sum b_k c_k.$$

From here we reach the Fourier inversion formula in three steps. The first step is to account for complex values. In a space with imaginary (or complex) basis vectors, we take the complex conjugate of the second vector, making the dot product

$$b \cdot c = \sum b_k c_k^{\star}.$$

This change ensures that the dot product of a vector with itself, which should be a squared length, is real and positive. The second step is to account for the infinite dimensionality. In an infinite-dimensional space, the sum becomes an integral over time. The third step is to account for the lengths of the vectors. We would like the basis functions F_k to be unit vectors, i.e. to have unit length. So we would like $F_k \cdot F_k$ to be 1. So we put a factor of $1/T$ in front of the integral and define the infinite-dimensional dot product as

$$f \cdot F_k \equiv \frac{1}{T}\int_T f(t)F_k^{\star}(t)\,dt.$$

The basis functions are $F_k = e^{j\omega_k t}$, so the Fourier inversion formula

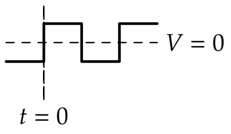
$$a_k = \frac{1}{T}\int_T f(t)e^{-j\omega_k t}\,dt,$$

says that

$$a_k = f \cdot F_k,$$

which is what you would find in a finite-dimensional space. [The minus sign in the exponent comes from the complex conjugation.]

18.1.2 Computing the coordinates (weights)



To compute the weight integral for the square wave, we need to choose where to put zero time and zero voltage. Since the circuit is time invariant, it does not matter where we put the origin of time. A choice that simplifies subsequent integrals is to take one of the pulses in the square wave and call its leading edge the time origin. It also does not matter where we put zero voltage, because we can measure all input and output voltages relative to that reference level. Use freedom to increase symmetry, and choose the most symmetric location for zero voltage, which is to place it halfway between the low and high levels of the square wave, and say that the high level is $V = 1/2$.

Exercise 80. The full argument about zero voltage is more subtle. Why, for this system, does it not matter where we place the zero voltage level?

It is also convenient to choose the integration range. Since f and $e^{-j\omega_k t}$ are periodic with period T, the integrand in their dot product is also periodic with period T. Therefore we can integrate over any convenient interval of length T. Use freedom to increase symmetry. For the square wave, integrating from $-T/2$ to $T/2$ is the most symmetric choice, and probably the least messy:

$$a_k = \frac{1}{2T}\left(\int_0^{T/2} - \int_{-T/2}^{0}\right)e^{-j\omega_k t}\,dt,$$

since $f(t) = 1/2$ from $t = 0$ to $T/2$, and $f(t) = -1/2$ from $t = -T/2$ to 0. Now do the integral:

$$a_k = -\frac{1}{j2T\omega_k}\left((e^{-j\omega_k T/2} - 1) - (1 - e^{j\omega_k T/2})\right).$$

Since $w_k T = 2\pi k$, this mess simplifies to

$$a_k = \frac{1}{j2\pi k}\left(1 - e^{-j\pi k}\right) = \begin{cases} \dfrac{1}{j\pi k} & (k \text{ odd}) \\ 0 & (\text{otherwise}). \end{cases}$$

The Fourier representation is

$$\text{square wave} = \sum_{\text{odd } k} \frac{e^{j\omega_k t}}{j\pi k}.$$

This sum looks complex (in the sense of having an imaginary part). However, the square wave is real. So the coefficients conspire to make the sum real. To see how, pair up corresponding terms.

For example, the $k = 5$ term is $a_5 e^{j\omega_5 t}$. Its partner is the $k = -5$ term, which is $a_{-5}e^{j\omega_{-5}t}$. Since a_{-5} is the complex conjugate of a_5, and $e^{j\omega_5 t}$ is the complex conjugate of $e^{j\omega_{-5}t}$, the product $a_{-5}e^{j\omega_{-5}t}$ is the complex conjugate of $a_5 e^{j\omega_5 t}$. So

$$a_{-5}e^{j\omega_{-5}t} + a_5 e^{j\omega_5 t} = 2\,\mathrm{Re}\,a_5 e^{j\omega_5 t}.$$

With $a_k = 1/(j\pi k)$, the sum of the paired terms is, for general k:

$$\frac{2\sin(2\pi kt/T)}{\pi k}.$$

So

$$\text{square wave} = \sum_{k\text{ odd, positive}}^{\infty} \frac{2\sin(2\pi kt/T)}{\pi k}.$$

Should the Fourier representation contain only sines? To decide, think about symmetry. With our choice of origin, the square wave is antisymmetric, meaning that $f(t) = -f(-t)$. So it should be composed only out of antisymmetric functions. Since sines are antisymmetric and cosines are symmetric, the square wave should indeed contain only sines.

Now that we have the Fourier representation for the square wave, we have a new method to find the output signal: See what the circuit does to each Fourier component, and figure out what signal has that representation. We do so for the two extremes of the input: a fast and a slow square wave.

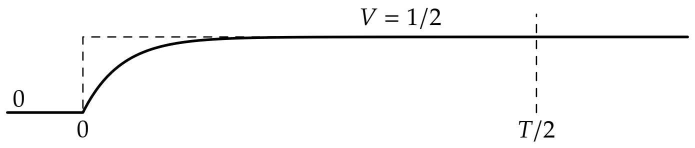
18.2 Slow square wave

The slow square wave is easier to analyze than is the fast square wave, so analyze the slow case first. 'Slow' has only a relative meaning: slow compared to another time. The only other time in this problem is the time constant τ of the RC circuit. So if the period T is long compared to τ, then the square wave is slow. We use the time representation to reason out the response to a slow square wave, then confirm the result with the Fourier representation.

18.2.1 Time representation

A true square wave starts at $t = -\infty$. But to start the time-representation analysis, instead imagine that the input signal is 0 for $t < 0$, and then turn on the square-wave signal at $t = 0$. Eventually the transient produced by turning on the input at $t = 0$ will settle down, but even before then, we can understand the main features of the output.

The first segment of the square-wave input is a positive voltage until $t = T/2$. So the RC circuit first sees a positive step that lasts for a time $T/2$. In the slow-square-wave extreme, the period T is much larger than τ. So as far as the RC circuit is concerned, the first segment of the square wave lasts forever, and the capacitor has a lot of time to charge to its steady-state value of $V = 1/2$. The output signal is an exponential approach to $1/2$:



After many RC time constants, the second segment of the square wave arrives. Relative to the first segment, it is a negative step that discharges the capacitor toward $-1/2$. The output voltage gets most of the way to $-1/2$ in a few RC time constants. Eventually the third segment of the square wave arrives and the RC adjusts to the new voltage level long before the fourth segment arrives.

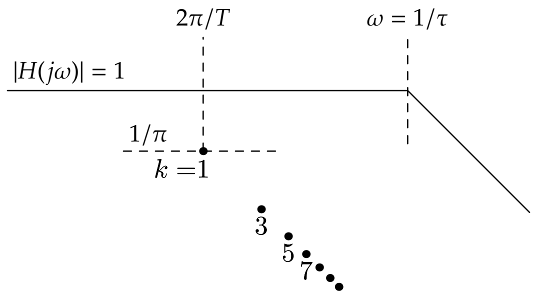
Except for a short interval after each transition, the output voltage tracks the input voltage almost exactly, meaning that the RC circuit acts like a wire. Let's see if we can understand that result using Fourier analysis.

18.2.2 Fourier representation of filtering

In the Fourier representation, the operation of the RC circuit is particularly simple. The square wave is a weighted sum of complex exponentials (or of sines), and the RC circuit merely adjusts the weights. The Fourier coefficient a'_k of the output signal is

$$a'_k = a_k H(j\omega).$$

To picture this product, the Bode plot for an RC circuit is helpful because its logarithmic magnitude axis converts multiplication into addition. So, on the Bode magnitude plot overlay the Fourier-coefficient magnitudes $|a_k|$, associating each one with its frequency $\omega_k = 2\pi k/T$. Here is a picture with $T = 100\tau$ showing the first several coefficients as dots:



The first dots lie in the frequency region of the unity-gain asymptote, so their magnitudes are unscathed by the RC circuit. Their phase is also untouched as long as their frequency is much less than $1/\tau$, because the low-frequency phase asymptote is $0°$.

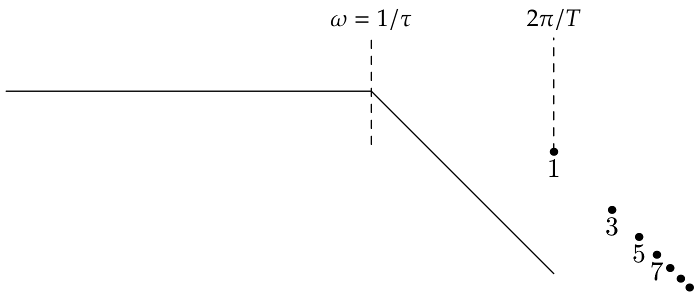
The higher-frequency Fourier components lie in the frequency region of the downward-sloping magnitude asymptote. So the RC circuit shrinks the high-frequency components. However, their amplitudes started small compared to the amplitudes of the low-frequency components, and the RC circuit makes these small amplitudes even tinier. These small changes affect the signal but not greatly, so the output signal is almost a replica of the square wave. However, the sharp edges are not replicated, The reason is that discontinuities require high (actually infinite) frequency, and the filtering squashes those high frequencies, so the sharp edges become smooth. These features are reflected in the output signal derived using the time representation.

18.3 Fast square wave

The other extreme of the input signal is a fast square wave whose period T is much smaller than the RC time constant τ. We will use the Fourier representation to predict the output, and leave you to confirm the result using the time representation (for example, by solving differential equations).

18.3.1 Fourier representation of filtering

A fast square wave contains the frequencies $\omega_k = 2\pi k/T$ (for odd k) but now the period T is very short, which means means $T \ll \tau$. So a fast square wave, reasonably enough, contains high frequencies. Here is the RC Bode magnitude picture overlaid with the Fourier amplitudes:



The dots look as they did for the slow square wave except that they are shifted to the right. This pictorial invariance is one reason to use a logarithmic frequency scale. The component frequencies are in the ratio $1 : 3 : 5 : \cdots$, no matter the period. The period determines the fundamental frequency (labelled with a '1'), but not those ratios. On a logarithmic scale, those ratios turn into relative distances, so the relative distances – the pattern – is independent of period. Changing the period just shifts the pattern.

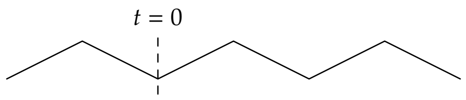
In their new position, the Fourier coefficients (the dots) are significantly altered by the RC circuit. The square-wave frequencies live in the frequency region of the downward-sloping asymptote rather than of the flat asymptote. The downward-sloping asymptote has a -1 slope, so the RC filter contributes a magnitude proportional to $1/\omega$. Since $\omega \sim k$, where k is the index of the Fourier basis function F_k, the RC filter contributes a factor proportional to $1/k$. So

$$|a_k'| \sim \underbrace{\frac{1}{\pi k}}_{|a_k|} \times \frac{1}{k} \sim \frac{1}{k^2} \qquad \text{(odd } k\text{)}.$$

To find the output signal from the Fourier coefficients, we also need to know the phase of a_k'. For a sufficiently fast square wave, all the dots lie in the region of the $-90°$ phase asymptote, which begins at $\omega = 10/\tau$. A $-90°$ phase means a factor of $1/j$. So the output Fourier coefficients pick up a factor of $1/j$, giving

$$a_k' \sim \underbrace{\frac{1}{j\pi k}}_{a_k} \times \underbrace{\frac{1}{k}}_{\text{mag.}} \times \underbrace{\frac{1}{j}}_{\text{phase}} = -\frac{1}{\pi k^2} \qquad \text{(odd } k\text{)}.$$

To find the resulting output signal we have to go the other direction: from the Fourier coefficients $a_k \sim 1/k^2$ to the function of time with those coefficients. That problem is hard in general, but in this case the solution is given in lecture. The $1/k^2$ coefficients (for odd k) produce a triangle wave:



Exercise 81. We were incomplete when we said that the downward-sloping asymptote contributed a magnitude factor proportional to $1/k$. Include the missing constant of proportionality to get the output signal's Fourier coefficients as a function of T (in the $T \ll \tau$ limit). Then find the amplitude of the resulting triangle wave as a function of T.

Exercise 82. Confirm the result of this section – that the output is a triangle wave – by analyzing the circuit in the time representation (again for $T \ll \tau$), perhaps by approximating the differential equation in that limit or by thinking about how the circuit behaves for short times.

18.4 Summary

Now you know another representation of signals: the Fourier representation. And you know another representation of systems: as filters that alter Fourier coefficients.

Exercise 83. What happens in the intermediate case, where the period T and the time constant τ are comparable? In particular, what happens in the case where the first few dots lie in the region of the unity-gain asymptote but in the region of the sloping phase asymptote?

Recitation 19
Fourier transform

After studying this chapter, you should be able to:

- derive the analysis and synthesis equations for Fourier transforms from their counterparts for relations for Fourier series;
- simplify convolution problems by using the frequency representation;
- find the DC level of a signal from the frequency representation; and
- find the frequency representation of periodic and sampled signals.

19.1 From Fourier series

The Fourier-transform representation generalizes the Fourier-series representation (**Chapter 18**). The Fourier-series representation of a signal with period T is

$$f(t) = \sum_{k=-\infty}^{\infty} f_k e^{j\omega_k t} \qquad \text{(Fourier-series synthesis equation)},$$

where $\omega_k = 2\pi k/T$ and the coefficients f_k are

$$f_k = \frac{1}{T}\int_T f(t)e^{-j\omega_k t}\,dt \qquad \text{(Fourier-series analysis equation)}.$$

Periodic signals are an important but limited category of signals. The Fourier transform removes this limitation. In the $T \to \infty$ limit, most of the coefficients f_k would become zero because of the factor of $1/T$. So we define a Fourier-transform coefficient $f(\omega_k)$ by multiplying f_k by T:

$$f(\omega_k) = T f_k = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt \qquad \text{(Fourier-transform analysis equation)}.$$

Now take the infinite-period limit ($T \to \infty$) in the Fourier-series synthesis equation. First replace f_k by $f(\omega_k)/T$ to get

$$f(t) = \sum_{k=-\infty}^{\infty} \frac{f(\omega_k)}{T} e^{j\omega_k t}.$$

Now take the continuum limit and turn the sum into an integral:

$$f(t) \approx \int_{-\infty}^{\infty} \frac{f(\omega_k)}{T} e^{j\omega_k t}\,dk.$$

Move the factor of $1/T$ into the dk:

$$f(t) \approx \int_{-\infty}^{\infty} f(\omega_k) e^{j\omega_k t}\, d\left(\frac{k}{T}\right).$$

As $T \to \infty$ and $d(k/T)$ goes to zero, replacing the sum by the integral becomes exact. Since $\omega = 2\pi k/T$, the $d(k/T)$ factor becomes $d\omega/2\pi$. Then

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(\omega) e^{j\omega t}\, d\omega \qquad \text{(Fourier-transform synthesis equation).}$$

This equation is the synthesis equation for the Fourier transform.

19.2 Convolution turns into multiplication

The transform or frequency representation $f(\omega)$ is useful for many reasons. Most importantly, it turns the difficult operation of convolution into the simpler operation of multiplication. It inherits this property – and most others – from Fourier series. We get a similar benefit from the logarithmic representation, which turns the hard operation of multiplication into the easy operation of addition.

You might wonder why one would care about the difficult operation of convolution. If the difficult operation is not useful, then simplifying it would also not be useful. But convolution is what a system does to a signal when the system is represented by its impulse response, so convolution is important for understanding what happens to signals.

For example, we could compute the impulse response of four identical RC filters (connected with buffers) directly in the time representation using convolution. Assuming that $\tau = 1$, each impulse response is $y(t) = e^{-t}$ (for $t \geq 0$). So the impulse response of the quadruple cascade is

$$y_4 = \delta \star y \star y \star y \star y.$$

What messy integrals that calculation would produce! Alternatively, we can use the frequency representation. The frequency representation of the delta function is 1 and of $y(t)$ is $y(\omega) = 1/(1 + j\omega)$. So the frequency representation of y_4 is

$$y_4(\omega) = \frac{1}{(1 + j\omega)^4}.$$

This frequency representation corresponds to a quadruple pole at $s = -1$, whose impulse response is $t^3 e^{-t}/6$ (for $t \geq 0$). This calculation done mostly in the frequency representation is simpler than the time-representation calculation that requires a quadruple convolution.

Exercise 84. Demonstrate the convolution property for Fourier transforms using the analysis and synthesis equations.

Exercise 85. Show that the response of the quadruple pole is $t^3 e^{-t}/6$ (for $t \geq 0$).

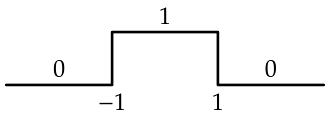
19.3 DC value and area

A particularly important characteristic of a signal is its average value, also known as its DC level. A related characteristic is the area under the signal. If the DC level is nonzero, then the area is infinite. If the DC level is zero, then the area is finite. The frequency representation makes it easy to find DC levels or areas and to find how a system alters them.

The area under $f(t)$ is $\int_{-\infty}^{\infty} f(t)\,dt$, which is just the frequency response at $\omega = 0$:

$$\begin{aligned} f(\omega = 0) &= \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt\bigg|_{\omega=0} \\ &= \int_{-\infty}^{\infty} f(t)\,dt. \end{aligned}$$

So the frequency representation contains the area as one of its points. To check this result, look at the frequency representation of a pulse. Being lazy, we choose the same pulse as in lecture:



It has area 2. Its frequency representation is

$$f(\omega) = \int_{-1}^{1} e^{-j\omega t}\,dt = 2\,\frac{\sin\omega}{\omega}.$$

And $f(\omega = 0) = 2$, which matches the area under the pulse.

What happens when the area is infinite? A simple example is the constant function $f(t) = 1$. Then $f(\omega) = 2\pi\delta(\omega)$, as you show in the homework. So an infinite area turns into an infinite spike at $\omega = 0$. For $f(t) = 1$, the DC level is 1 and the area under the spike is 2π. Perhaps this relation between the DC level and the spike area is a general one:

$$\text{DC level} = \frac{1}{2\pi} \times \text{area under the spike at } \omega = 0.$$

To check this conjecture, compute the area under the spike by integrating just in its neighborhood:

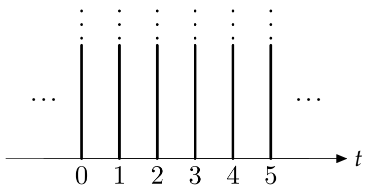
$$\begin{aligned} \text{area under the spike at } \omega = 0 \quad &= \lim_{\epsilon\to 0} \int_{-\epsilon}^{\epsilon} f(\omega)\,d\omega \\ &= \lim_{\epsilon\to 0} \int_{-\epsilon}^{\epsilon} f(\omega)e^{-j\omega t}\,d\omega. \end{aligned}$$

The magic of inserting $e^{-j\omega t}$ is justified because ω ranges between $-\epsilon$ and ϵ making $e^{-j\omega t} \approx 1$. The last integral is the Fourier inverse of $2\pi f(\omega)$ after throwing away all frequencies except those near $\omega = 0$. In the limit of $\epsilon \to 0$, only the zero or DC frequency remains. The resulting Fourier inverse produces 2π times the DC part of $f(t)$, which confirms the preceding conjecture about the DC level.

Exercise 86. What is the DC level of $f(t) = \delta(t)$? Check that result against $f(\omega = 0)$.

19.4 Representation of periodic signals

We introduced the Fourier transform as generalizing Fourier series. Since Fourier series represent periodic signals, the Fourier transform should also represent periodic signals. To see how it does so, try the simplest periodic signal: a periodic train of spikes that has no beginning and no end. For simplicity, take the spike interval to be 1. Then



$$f(t) = \sum_{n=-\infty}^{\infty} \delta(t+n).$$

Pause to try 61. Find $f(\omega)$.

Then $f(\omega)$ is

$$f(\omega) = \int_{-\infty}^{\infty} \left(\sum_{n=-\infty}^{\infty} \delta(t+n) \right) e^{-j\omega t}\, dt.$$

Now swap integration and summation, not worrying whether that switch is legal, and then integrate out the delta function:

$$\begin{aligned} f(\omega) &= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t+n) e^{-j\omega t}\, dt \\ &= \sum_{n=-\infty}^{\infty} e^{jn\omega}. \end{aligned}$$

Since each term $e^{jn\omega}$ has period 2π, the sum has period 2π. So we need to figure out the sum only over a range of length 2π. A convenient range is $\omega = [-\pi, \pi]$. The easiest value is $\omega = 0$ because each term is 1. Since there are an infinite number of terms, $f(\omega = 0) = \infty$. For other values of ω, the easiest method is again to close ones eyes to lack of rigor. The sum is a doubly infinite geometric sequence:

$$\sum_{n=-\infty}^{\infty} e^{jn\omega} = \underbrace{\sum_{n=-\infty}^{-1} e^{jn\omega}}_{\text{first half}} + \underbrace{\sum_{n=0}^{\infty} e^{jn\omega}}_{\text{second half}}.$$

Each half is a geometric series with ratio r or $1/r$, where $r = e^{j\omega}$. The first half is then

$$\frac{r^{-1}}{1-r^{-1}} = \frac{-1}{1-r}.$$

. The second half is $1/(1-r)$. The sum of the two halves is zero! So the original sum for $f(\omega)$ is infinite at $\omega = 0$ and is zero elsewhere (for $\omega = [-\pi, \pi]$). The combination being infinitely high at only one ω and being zero elsewhere suggests that

$$\sum_{n=-\infty}^{\infty} e^{jn\omega} \sim \delta(\omega).$$

The missing constant of proportionality is 2π, so

$$f(\omega) = \sum_{n=-\infty}^{\infty} e^{jn\omega} = 2\pi\delta(\omega) \qquad (\omega = -\pi \ldots \pi).$$

Since $f(\omega)$ is periodic, the entire $f(\omega)$ is

$$f(\omega) = 2\pi \sum_{k=-\infty}^{\infty} \delta(\omega - 2\pi k).$$

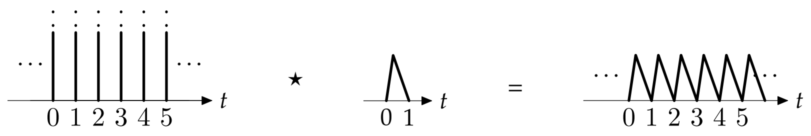
Exercise 87. What is the DC level of the original train of impulses? Use the DC level to verify the factor of 2π.

Exercise 88. [hard] By integrating the sum for $f(\omega)$, show that $f(\omega) = 2\pi\delta(\omega)$.

Exercise 89. Showing that $f(\omega) = 0$ for $\omega \neq 0$ used dubious operations, although the result is correct. Justify the dubious operations.

Exercise 90. Now imagine that the spikes are separated by π rather than by 1 time unit. What is $f(\omega)$?

A signal $f(t)$ that is more general than a train of spikes also has a convenient frequency representation. Any periodic signal is the convolution of one period $f_1(t)$ with a spike train with that period. For example,



Convolution of time representations is multiplication of frequency representations, so

$$f(\omega) = f_1(\omega) \times 2\pi \sum_{k=-\infty}^{\infty} \delta(\omega - 2\pi k).$$

So a signal whose time representation is periodic has a frequency representation composed of regularly spaced spikes. The spacing of the spikes is 2π (or $2\pi/T$ when the period is not 1) and their amplitude is given by the frequency representation of one period $f_1(t)$. The frequency representation of a periodic signal is therefore a sampled version of the frequency representation of one period.

19.5 Sampled signals and duality

It is interesting that the transform of a spike train is another spike train, and that the inverse transform of a spike train is another spike train. This property is an example of duality: that changing from the time representation to the frequency representation is almost the same operation as changing in the opposite direction.

Compare the forward and backward forms:

$$\begin{aligned} f(\omega) &= \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt; \\ f(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} f(\omega)e^{j\omega t}\,d\omega. \end{aligned}$$

The structures of the two formulas differ only by the presence or absence of a minus sign in the exponent and of a factor of $1/2\pi$. Therefore, properties of the transform that do not depend on these details apply in both directions.

An important example is the convolution property: Convolution of time representations turns into multiplication of frequency representations. Its derivation did not depend on our sign convention for ω or on our convention about where to put the 2π. So the convolution property works in the reverse direction, meaning that convolution of frequency representations turns into multiplication of time representations.

An example is the frequency representation of sampled or discrete-time signals. One way to make a discrete-time signal $f[n]$ is to sample a continuous-time signal $f(t)$. To sample a signal, multiply it by the train of delta functions to produce a train of spikes. The spikes are each delta functions whose amplitudes are modulated by $f(t)$. Multiplication of these time representations therefore convolves $f(\omega)$ with the frequency representation of the spike train. The spike train in time is represented by a spike train in frequency. So the frequency representation of $f[n]$ is the convolution of $f(\omega)$ with a spike train. Thus, the discrete-time signal $f[n]$ has a periodic frequency representation.

This periodicity is built into the so-called discrete-time Fourier transform, which lives on the unit circle in the z plane instead of on the infinite-extent ω axis. Placing the transform on the unit circle enforces periodicity. This transform is the subject of the next chapter.

Recitation 20
Fourier series as rotations

After studying this chapter, you should be able to:

- explain how converting to and from the Fourier representation is a rotation;
- deduce properties of Fourier series based on this geometric interpretation; and
- apply Parseval's theorem to do interesting sums.

20.1 Why study Fourier series

Much of our knowledge of the world is contained in functions. The voltage produced by a microphone, as a function of time, represents music that can be encoded digitally and stored on a compact disc. The position of a planet, as a function of time, represents an orbit. Functions are everywhere. They are also extremely complex objects. Any sufficiently complex object is too difficult to comprehend in one glance because our minds are limited. We therefore study functions from many vantage points by using multiple representations.

Transforms, such the Fourier and Laplace transforms, are ways to change representation. Which representation you use depends on which questions you want to answer. A representation may be ideal for answering one question but lousy for answering another question. Build a library of representations – of tools to analyze functions – and know when and how to use each tool.

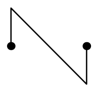
The purpose of this chapter is to show, using many examples, how Fourier series are a change of representation, and how that change is like a rotation of axes. Another purpose is to illustrate several ideas and skills useful for understanding many mathematical and scientific concepts:

- **Discretization**. Discretized (or lumped) systems are simpler to understand than fully general, continuous systems. For example, in analyzing a circuit, we may base it on the inductor, capacitor, or resistor values in the circuit. But Maxwell's equations of electromagnetism know nothing of inductors, capacitors, or resistors. But we can approximate the complexity of Maxwell's equations by dividing the system into discrete lumps: capacitors, resistors, and inductors. A discrete system can be analyzed with ordinary differential equations, whereas a continuous system requires a partial differential equations (much harder!).
- **Continuity**. Although continuous systems are hard to understand, continuous changes help you understand a system. The idea of continuity is that small changes to a system should produce only small changes to its behavior.
- **Representation**. If a picture is worth a thousand words, a representation is worth a thousand pictures. To be a skilled analyst and designer of systems, become fluent in many representations and how to convert among them.

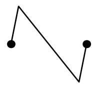
20.2 Making a representation

Before studying how to *change* representations, first create a function to be changed. Imagine all possible functions $g(t)$, where $0 \le t \le 1$. The set of all functions is even larger than this limited class, but this limited class is already too big. So study an even more limited class: functions where $g(0) = g(1) = 0$. These functions are the possible shapes of a unit-length guitar or violin string. The ideas developed in studying these functions work for the shapes and behavior of longer strings, of a membrane (a drum head), or of analog filters (for example a stereo amplifier).

The function that we use throughout the example is one tooth of a sawtooth wave, with the endpoints marked with dots. You might feel cheated because this function is not really 0 at its endpoints. Instead it has a discontinuity.



If that discontinuity bothers you, and it should, imagine a similar function with the infinitely steep edge replaced by a very steep edge – and then make that edge steeper and steeper until it gets infinitely steep. This steepening argument uses the principle of continuity. From here on, we analyze the infinitely steep, pure sawtooth.

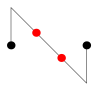


This restricted set of functions, of which the sawtooth is merely one, forms a vast collection. To shrink the collection to a manageable size, sample each function at equally spaced times. Sampling represents a continuous function by a finite set of values. Each function then becomes a point in a finite-dimensional space. By discretizing the continuous system, we simplify our analysis and make it easier to picture operations such as converting to the Fourier representation.

To illustrate the ideas in the simplest possible case, sample the function first at only one point (other than the fixed endpoints). The functions are represented by their value at $t = 1/2$, so each function is represented by one number: $g(1/2)$. Equivalently, each function is represented by a point on a single axis. With only one axis, there is no possibility to rotate the axes. To illustrate interesting transformations, one dimension or sample is *too* simple.

20.3 Two-sample representation

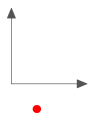
Therefore try $p = 2$ samples, where p is the number of samples. The eventual plan is to increase p to ∞ to recover the original function. With $p = 2$, specifying a function requires two numbers: $g(1/3)$ and $g(2/3)$. A function becomes a point or a vector in a two-dimensional space. Here is the sawtooth function f sampled at two points (other than the easy endpoints). For comparison, the unsampled sawtooth function is shown lightly in the background.

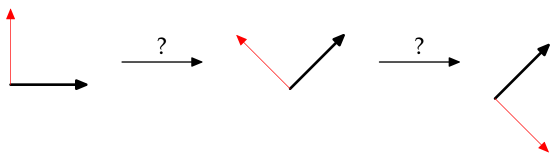


This drawing becomes a point in a two-dimensional space where $g(1/3)$ is the coordinate on the horizontal axis and $g(2/3)$ is the vertical coordinate.

Pause to try 62. What are $g(1/3)$ and $g(2/3)$?

Since $g(t) = 1 - 2t$, the samples are $g(1/3) = 1/3$ and $g(2/3) = -1/3$. Therefore in this two-dimensional space, the sampled function has coordinates $(1/3, -1/3)$. Changing representation means using new coordinate axes. The function is still the same: It is the same point. But its coordinates change when the axes change. The next problem gives you practice with two-dimension transformations.





Pause to try 63. In the above change of axes, what transformations take you from the first to the second to the third (final) set of axes?

In this sequence, the first transform is a rotation by 45 degrees counterclockwise, and the second is a sign flip of the rotated second axis. The resulting coordinate system is the two-dimensional Fourier coordinate system. Looking ahead, it is a modified Fourier representation whose basis functions are sines and cosines, rather than complex exponentials. Using complex exponentials instead is more efficient when one computes Fourier series. However, using sines (and cosines) eliminates the complex-number manipulations, and is more useful for sketching basis functions and therefore for understanding the concepts of Fourier series. The focus of this chapter is the concepts, so it chooses the sines and cosines.

The chosen coordinate system has several important properties that we investigate shortly. But first a few questions for you to check your understanding.

Pause to try 64. In the new coordinate system, what are the coordinates of the sampled sawtooth function? These coordinates are different from its original coordinates $(1/3, -1/3)$. If the function remains the same, how can its coordinates change?

The new coordinates are $(0, \sqrt{2}/3)$. They are indeed different from the original coordinates $(1/3, -1/3)$. The function is still the same (sampled) sawtooth and therefore the same point in function space. However, its coordinates changed because *the axes (the representation) has changed underneath it.* We repeat this point because it is so important.

Almost any change of representation changes coordinates. The particular change is among a special class of changes with useful properties. Let's look at those properties, because they remain when we sample at more points and, eventually, when we build the general Fourier representation based on an infinite number of samples.

To see the first useful property, try the following:

Pause to try 65. How long is each basis vector (colloquially speaking, each axis)?

There are several ways to answer this question. First, find the coordinates of the new basis vectors *using the old representation.* The old $(1, 0)$ basis vector becomes, after the 45-degree rotation, $(1/\sqrt{2}, 1/\sqrt{2})$ and remains the same after the flip because the flip affects only the rotated $(0, 1)$ axis. The length of $(1/\sqrt{2}, 1/\sqrt{2})$ is one. The old $(0, 1)$ basis vector becomes, after the 45-degree rotation, $(-1/\sqrt{2}, 1/\sqrt{2})$, and becomes, after the flip, $(1/\sqrt{2}, -1/\sqrt{2})$. It too has length one.

Alternatively and more elegantly, look at the sequence of transforms – a rotation and a flip – that produced the new axes. The first transform, a rotation, does not change lengths. The second transform, a flip, also does not change lengths. So the new axes have the same lengths (one) as the original axes.

The next property is illustrated in this question:

Pause to try 66. What is the angle between the basis vectors?

Neither transform (a rotation or a flip) changes angles, so each basis vector is perpendicular to the other. The new coordinate system, like the old, is therefore **orthogonal**. Since the basis vectors also have unit length, the new coordinate system, like the old, is also an **orthonormal** coordinate system.

An amazing property, which is a consequence of the previous ones, is the length of the vector representing the function. In the original representation the vector is $(1/3, -1/3)$.

Pause to try 67. What is the length of that vector?

Here is the calculation for the original representation:

$$\text{length of } (1/3, -1/3) = \sqrt{(1/3)^2 + (-1/3)^2} = \sqrt{2/9} = \sqrt{2}/3.$$

In the new representation, the vector is $(0, \sqrt{2}/3)$.

Pause to try 68. What is its length?

In the new representation, the length is just the second coordinate since the first coordinate is zero. So the lengths in the two representations are both $\sqrt{2}/3$.

Pause to try 69. Explain why *all* lengths are unchanged by our change of axes, not just the length of the particular function we are using (the sawtooth function).

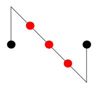
Length preservation may seem trivial, which is why we introduce it first in two dimensions (i.e. with two samples). When we generalize to an infinite number of samples, this seemingly trivial property turns into **Parseval's theorem**.

A further property of an orthonormal change of representation is that any function represented in the old system (here, by the two values $g(1/3)$ and $g(2/3)$) has a representation in the new system. The new system is therefore **complete**: every function can be represented.

Completeness, like length preservation, may seem obvious when we use only two samples. However, it becomes nonobvious and important in the general case with an infinite number of samples. It then says that any function, with a few disclaimers, has a Fourier representation, namely as a sum of sines (and cosines). This property is counterintuitive. Even the supremely talented Euler, when faced with choosing whether to give up the principle of superposition for linear differential equations or to accept this completeness property, was ready to give up superposition! He said it was crazy to think that any function could be represented as a sum of sines (and cosines). If Euler finds a concept crazy, we too can expect to struggle with the idea.

20.4 Three-sample representation

These same properties and patterns occur with $p = 3$ samples, although the transformations are harder to visualize because three dimensions are harder to draw than two dimensions (which is why we spent so much time investigating two dimensions, i.e. with $p = 2$.). In this three-sample world, a function can be represented as three numbers: $g(1/4)$, $g(2/4)$, and $g(3/4)$.



Pause to try 70. Why did I not include $g(0)$ and $g(1)$ in the representation? What are the coordinates of the sampled sawtooth in this representation?

Here are the three new basis vectors, chosen for the moment by fiat, shown using their coordinates in the old representation:

$$v_1 = \begin{pmatrix} 1/2 \\ 1/\sqrt{2} \\ 1/2 \end{pmatrix}, \quad v_2 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad v_3 = \begin{pmatrix} 1/2 \\ -1/\sqrt{2} \\ 1/2 \end{pmatrix}.$$

Pause to try 71. What is the length of each basis vector v_k? What are the six angles between each pair of vectors?

As you've shown, the basis vectors are orthogonal (each vector is perpendicular to the others) and normalized (each vector has unit length), so they form an orthonormal coordinate system.

We want to represent the sawtooth function in this coordinate system, given its coordinates in the usual representation, which are $(1/2, 0, -1/2)$.

Pause to try 72. What are its coordinates in the new representation?

In an orthonormal coordinate system, it is easy to find the coordinates of a vector. Take the vector (i.e. the point or the function) and extract its component along each new axis using the dot product:

$$k\text{th component} = g \cdot (\text{basis vector } k).$$

The dot product tells you the overlap between two functions. Here, it tells you the overlap between the candidate function g and the kth basis vector – or, what is the same object, the kth basis function. The sampled sawtooth sits on the second axis in the new representation, so its first and third coordinates are 0 in the new representation. Here is the computation of the second coordinate:

$$\text{2nd coordinate} = \underbrace{\begin{pmatrix} 1/2 \\ 0 \\ -1/2 \end{pmatrix}}_{g} \cdot \underbrace{\begin{pmatrix} 1/\sqrt{2} \\ 0 \\ -1/\sqrt{2} \end{pmatrix}}_{v_k} = \frac{1}{\sqrt{2}}.$$

So the sampled sawtooth, in the new representation, is represented by these coordinates

$$g = \begin{pmatrix} 0 \\ 1/\sqrt{2} \\ 0 \end{pmatrix}.$$

The three components are the three dot products, each with respect to one basis vector. Each dot product measures how close the represented function g is to each basis vector, a.k.a. basis function. Remember that a vector, including a basis vector, is a function: Points, or vectors, are how we represent functions. So the three dot products give three pieces of information: how close g is to each basis function. Those three pieces of information locate g in function space by a process akin to triangulation that is used to locate earthquakes. They are located by measuring their distance from several (typically, three) seismographs, and then finding the one point, the source, that is has the correct distance from each seismograph. In function space, the three distances (the three dot products) uniquely determine the function.

Pause to try 73. What are the lengths of g in the old and new representations?

As you've just shown, the length of the sawtooth function is the same ($1/\sqrt{2}$) in the old and new representations.

20.5 Many-sample representation

Now return to the original space of functions before they got sampled at only a few points, and let's make p large. But before doing that, here's the recipe that I used to choose the basis vectors. Take the function $\sin k\pi t$, where $k = 1 \ldots p$, and sample it at p equally spaced points:

$$t = \frac{1}{p+1}, \frac{2}{p+1}, \frac{3}{p+1}, \ldots, \frac{p}{p+1}.$$

The values of g at these p samples produce the *unnormalized* basis vector. Then normalize it (make it have unit length) to get the vector that I used.

Pause to try 74. Check that the v_k vectors (for $p = 3$) result using the preceding recipe.

To find the kth new coordinate, use the same recipe as when $p = 3$: Take the dot product of the normalized basis vector (or point or basis function) with the example function g.

As p goes to infinity, nothing essential changes. However, one issue needs to be dealt with carefully, which is the normalization.

Pause to try 75. Show that the new, *unnormalized* vectors produced by the above recipe have length $\sqrt{(p+1)/2}$. [Difficult unless you are fluent with series!]

As p goes to ∞, the unnormalized length goes to ∞. Normalizing the vector means dividing by the unnormalized length, which makes every component zero! So our results would be boring, often reducing to the grand conclusion that $0 = 0$.

This length problem is solved by slightly adjusting the definition of the dot product.

Pause to try 76. How could fixing the dot product solve a problem with length? In other words, what do length and dot product have to do with each other?

For finite p, the dot product means doing componentwise multiplication and then summing the results. That definition is fine for finite p. However, with an infinite number of terms in the sum, which happens when $p = \infty$, it is natural to replace the sum by an integral. We therefore define the dot product of two functions $g(t)$ and $h(t)$ by

$$g \cdot h = \int_0^1 g(t)h(t)\,dt.$$

Especially in quantum chemistry, such an integral is often called the *overlap integral*. The term overlap arises because functions that vary together have a positive dot product; functions that vary opposite to each other have a negative dot product; and functions that vary randomly (have no overlap) with respect to each other have zero dot product.

The length of an unnormalized basis function u_k is $\sqrt{u_k \cdot u_k}$, and the dot product inside the square root is $\int_0^1 u_k(t)u_k(t)\,dt$, which does not blow up. With $u_k = \sin k\pi t$ the integral is $1/2$.

Pause to try 77. Show that

$$\int_0^1 \sin^2 k\pi t\,dt = \frac{1}{2}.$$

Using this definition of dot product, the coordinates of the sawtooth g become

$$f_k \equiv k\text{th component} = \int_0^1 g(t)\,\sqrt{2}\sin k\pi t\,dt,$$

where the $\sqrt{2}$ comes from normalizing the sin functions to have unit length. These f_k are the **Fourier coefficients** of g.

Exercise 91. Show that $f_k = 0$ for odd k and $2\sqrt{2}/k\pi$ for even k.

20.6 Parseval's theorem

Parseval's theorem relates the square of the original function and the squares of the Fourier coefficients. Go back to the two- or three-dimensional analyses to see where it comes from. In those analyses, the old and new vector have the same length because the change of representation is a rotation perhaps with a flip.

Let's look again at the squared lengths in the old and new representations, now with $p = \infty$. This limit produces the recipe for Fourier series of continuous-time functions.

We'll investigate the squared length rather than the length itself, in order to avoid many square-root signs. The squared length of a vector is the sum of squares of the components. In the before (usual) representation, the sum of squares is $\int_0^1 g(t)^2\,dt$. In the after (Fourier) representation, the sum of squares is the squared sum of the Fourier coefficients: $\sum_1^\infty (a_k)^2$. The equality of the sum and integral is Parseval's theorem:

$$\sum f_k^2 = \int_0^1 g(t)^2\,dt.$$

Pause to try 78. Given the geometric meaning (length) attached to the integral and sum, why are their values equal?

The change from the old to the new representation is a rotation (perhaps with a flip). Parseval's theorem restates, in an infinite-dimensional space, the geometric fact that **rotations do not change lengths**. That requirement is 99% of the definition of a rotation.

Let's check that the theorem works. As they used to say in arms control negotiations: trust but verify. To verify the Fourier-representation half of the theorem, we need to find the Fourier coefficients, which you should already have done in a preceding question and which I calculate now. They are:

$$f_k = g \cdot v_k = \int_0^1 g(t) \cdot \underbrace{\sqrt{2}\,\sin k\pi t}_{v_k}\,dt,$$

where $g(t) = 1 - 2t$. We have to compute one integral for each k. But some integrals vanish. For example, look at the integral for f_1. The function $g(t)$ is antisymmetric about $t = 1/2$, whereas the Fourier function $\sqrt{2}\sin \pi t$ is symmetric about $t = 1/2$. The integrand, as the product of these two functions, is antisymmetric about $t = 1/2$, which is the midpoint of the integration range. So the integral vanishes. A similar argument works for any odd k. So those f_k's are zero. *Moral:* The Fourier functions that are used have some symmetries from the original function. If the original function has reflection symmetry, then only those Fourier functions with reflection symmetry will be used.

When k is even, we have integrals to do. The integral splits into two parts, one part from each term in $1 - 2t$:

$$\int_0^1 \sqrt{2}\,\sin k\pi t\,dt - 2\int_0^1 t\sqrt{2}\,\sin k\pi t\,dt.$$

When k is even, the first integral is zero because $\sin k\pi t$ is antisymmetric about $t = 1/2$. So the only integral remaining is

$$-2\int_0^1 t\sqrt{2}\,\sin k\pi t\,dt,$$

and only for even k. Integrating by parts (or differentiating under the integral sign with respect to k) gives $2\sqrt{2}/k\pi$. That piece is the only contributor to the Fourier coefficient, so

$$f_k = \begin{cases} 2\sqrt{2}/k\pi, & k \text{ even};\\ 0, & k \text{ odd}. \end{cases}.$$

Now we can do the Parseval sum. It is

$$\sum_{k\text{ even}} f_k^2 = \sum_{k\text{ even}} \frac{8}{k^2\pi^2}.$$

Pause to try 79. Do the sum!

To do the sum, a useful result is

$$\sum_1^\infty \frac{1}{n^2} = \frac{\pi^2}{6},$$

a famous result derived by Euler. The sum for f_k^2 uses only the even terms, and

$$\sum_{n\text{ even}} \frac{1}{n^2} = \frac{\pi^2}{24}.$$

The factor of 8 in the Parseval sum, combined with this result, produces $\sum_{k\text{ even}} f_k^2 = 1/3$.

Let's hope that the other half of Parseval's theorem gives the same value. To do that half, square $g(t)$ and integrate:

$$|\text{length of } g \text{ in the usual rep}|^2 = \int_0^1 (1-2t)^2\,dt.$$

You can do this integral graphically. The function $(1-2t)^2$ is a parabola. A parabola occupies one-third of its bounding rectangle (which is the form in which Archimedes knew this result that we would usually do by integration), just as a triangle occupies one-half of its bounding rectangle. The bounding rectangle has unit area so the squared length of g is $1/3$, which is also the squared length of g as measured in the Fourier representation! Alternatively, we could have used the integral in the time representation to evaluate the famous sum

$$\sum_1^\infty \frac{1}{n^2} = \frac{\pi^2}{6}.$$

20.7 Why those functions?

Why did we use those particular basis functions? Nothing in this discussion seemed to depend on their being sines. What about using other basis functions? You can! Another set of basis functions would produce yet another function representation. For example, the Legendre polynomials or Bessel functions are another useful set of basis functions and generate useful representations.

To choose the representation, return to first principles: Use the representation that is most useful for answering the questions that you have. Sines (and cosines) are a useful choice because they arise naturally in the equations that describe the motion of springs, waves, membranes, strings, *LRC* circuits, and much else. They arise because Newton's second law has a second time derivative in it; because the net force on a piece of a stretched string depends on the local curvature, which incorporates a second space derivative; or because Maxwell's equations result in a wave equation, whose second derivatives produce sines and cosines.

Wherever the origin of the second derivative, it has the eigenvalue form

$$\text{second derivative of } f = \text{constant} \times f.$$

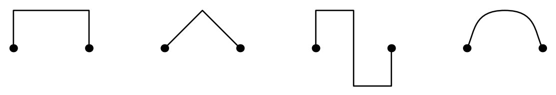
Exercise 92. Show that the net force on a piece of a stretched string is proportional to a second derivative.

Sines and cosines, or complex exponentials, satisfy this eigenvalue form. The benefit of representing functions in terms of sines and cosines is that the sine and cosine functions behave independently from one another. Each component does its own motion, at its own frequency, independent of the others. Speaking in linear-algebra terms, the Fourier basis functions diagonalize the second derivative operator. The full connection with linear algebra is a large topic, and you will see it again as you deepen your understanding of engineering systems.

20.8 Problems

Exercise 93. Add the explanation for $p = 4$ samples, using the same sawtooth g.

Exercise 94. Redo the analysis, perhaps including $p = 4$, using another function for g. Here are several candidates (in order): the upper half of a square wave, a triangle wave, a symmetric square wave, and a parabolic hump.



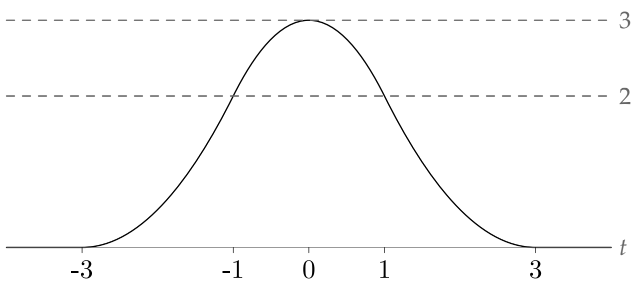
Exercise 95. Redo the analysis for functions over the domain $-\infty$ to ∞, instead of the limited domain 0 to 1 used here. This generalization results in **Fourier transforms**.

Recitation 21
A tour of Fourier representations

After studying this chapter, you should be able to:

- explain the connection between Fourier series and transforms in terms of sampling the frequency representation; and
- explain the connection between discrete-time and continuous-time Fourier representations in terms of sampling the time representation.

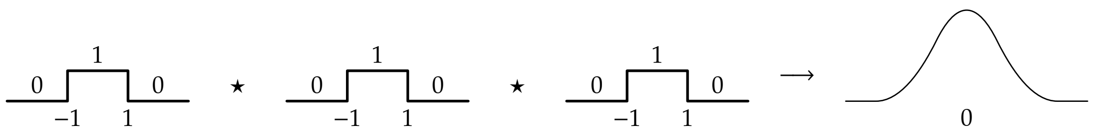
To explore the relations between the four Fourier representations, we use this continuous-time, aperiodic signal $f(t)$:



It looks like a segment of a squared cosine, but its exact time representation is

$$f(t) = \begin{cases} 0 & \text{for } t < -3; \\ (t+3)^2/2 & \text{for } -3 \le t \le -1; \\ 3 - t^2 & \text{for } -1 \le t \le 1; \\ (t-3)^2/2 & \text{for } 1 \le t \le 3; \\ 0 & \text{for } t > 3. \end{cases}$$

These equations look messy but they result from the simple recipe of convolving a pulse with itself twice:



To see that the shape of $f(t)$ is plausible, think about how convolution affects smoothness. The pulse itself has discontinuities. Convolving the pulse with itself is a form of low-pass filtering, so it produces a smoother signal: a triangle with no discontinuities. The triangle has slope discontinuities at its vertices. Convolving the triangle with the pulse low-pass filters the triangle, making a signal $f(t)$ with neither discontinuities nor slope discontinuities.

Exercise 96. Confirm that $f(t)$ has no slope discontinuities even at the two joints $t = -1$ and $t = 1$.

Exercise 97. Compute the triple convolution to confirm that $f(t)$ is indeed as claimed.

We use $f(t)$ to illustrate connections among:

1. the continuous-time Fourier transform (CTFT),
2. the continuous-time Fourier series (CTFS),
3. the discrete-time Fourier transform (DTFT), and
4. the discrete-time Fourier series (DTFS), usually implemented (and misnamed) as the fast Fourier transform (FFT).

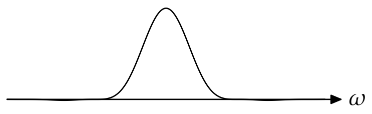
The continuous-time Fourier transform places no restrictions on the signal: It may or may not be periodic, and it may or may not be sampled. This $f(t)$ is general in that it is aperiodic and nonsampled. The other three transforms restrict the signal. as we examine each transform, we make a corresponding signal from $f(t)$ that meets the restrictions of that transform.

21.1 Continuous-time Fourier transform

The continuous-time Fourier transform of $f(t)$ is

$$f(\omega) = \left(2\frac{\sin\omega}{\omega}\right)^3 .$$

which looks like:



To check the transform, look at the DC value $f(\omega = 0)$. It is $f(\omega = 0) = 8$, which means that $f(t)$ should have area 8. And it does, as you can see from a convolution argument. The pulse has area 2, so the triple convolution of the pulse produces a function with area $2 \times 2 \times 2 = 8$.

Exercise 98. Confirm that the area under $f(t)$ is 8 by integrating $f(t)$.

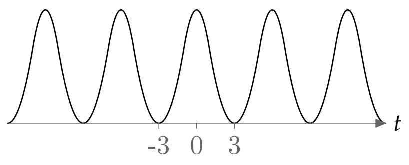
Exercise 99. Compute the Fourier transform to confirm that

$$f(\omega) = \left(2\frac{\sin\omega}{\omega}\right)^3 .$$

Hint: Use the convolution property instead of doing the Fourier transform integral directly.

21.2 Continuous-time Fourier series

A Fourier series represents only periodic signals, so let's make a periodic version of $f(t)$. There are ways to do so. Any periodic version of $f(t)$ will illustrate the connection between the Fourier transform and Fourier series. But the simplest periodic version is obtained by taking the nonzero region of $f(t)$ as the period; in mathematical jargon, we are using the *support* of $f(t)$ as the period. That region is $-3 < t < 3$, making the period $T = 6$. The resulting signal $f_p(t)$ looks like



We next find the Fourier transform of $f_p(t)$, which we can do for any signal. The transform will turn out almost identical to the Fourier series. To find the transform, divide and conquer. The periodic signal is the convolution of $f(t)$ with a comb – which is the fancy name for an impulse train. The comb's teeth are unit-area delta functions spaced 6 units apart.

Dividing $f_p(t)$ into a aperiodic part convolved with a comb makes finding its transform easy because convolution of time representations is equivalent to multiplication of frequency representations. So the Fourier transform of $f_p(t)$ is:

$$f_p(\omega) = f(\omega) \times \text{Fourier transform of the comb.}$$

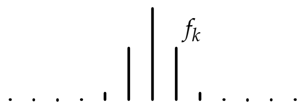
The Fourier transform of a comb with period $T = 6$ and amplitude 1 is a frequency comb with period $2\pi/T = \pi/3$ and amplitude $2\pi/T$. [The factor of 2π in the amplitude results from our convention for the inverse Fourier transform, and is the least important factor in this discussion.]

Multiplying any function by a comb samples that function. This process produces regularly spaced delta functions whose amplitudes (areas) are the values of the function at the comb locations. Therefore, the Fourier transform of the periodic function $f_p(t)$ is a sampled version of the Fourier transform of one period $f(t)$: *Making a function periodic samples its transform.*

The Fourier *series* represents the same information as the sampled transform $f_p(\omega)$, but represents it conveniently. Rather than using delta functions, which are hard to draw, the Fourier series uses their area directly, and just lists the areas indexed by an integer k (rather than as a function of frequency). As a bonus, the Fourier series drops an annoying factor of 2π. In terms of the transform of one period, the kth Fourier coefficient f_k is

$$f_k = \frac{f(\omega_k)}{2\pi} \quad \text{where } \omega_k = 2\pi k/T.$$

Those coefficients are illustrated in this diagram:



The tops of the samples sketch out the shape of $f(\omega)$.

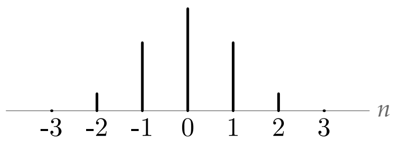
The coefficients f_k are of order $1/k^3$ for large k, reflecting the $1/\omega^3$ factor in the transform

$$f(\omega) = \left(2\frac{\sin\omega}{\omega}\right)^3.$$

So the time signal $f(t)$ and its periodic version $f_\text{p}(t)$ hardly use high-frequency oscillations, meaning that they are very smooth. You can understand the $1/k^3$ amplitude roll off by looking at levels of smoothness together with the Fourier coefficients for each level. An infinitely discontinuous signal like a delta function has spectrum $f_k \sim 1$, which means no roll off and infinite bandwidth. A finitely discontinuous signal – for example a pulse or a sawtooth – has spectrum $f_k \sim 1/k$. A continuous signal with slope discontinuities – for example a triangle – has spectrum $f_k \sim 1/k^2$. So the signal $f(t)$, which is one level smoother than the triangle, should and does have spectrum $f_k \sim 1/k^3$.

21.3 Discrete-time Fourier transform

Our next Fourier representation is the discrete-time Fourier transform. As its name suggests, it is most closely related to the continuous-time Fourier transform. So forget about Fourier series for the moment and return to the Fourier transform of $f(t)$. Rather than making $f(t)$ periodic, we now sample $f(t)$ to make a discrete-time signal. Sampling means multiplication by a comb, which produces the sampled signal composed of delta functions. You can sample using any comb spacing Δt, and each spacing (and starting position) generates its own discrete-time signal. For simplicity, we use a comb with unit spacing to get the sampled signal:



The spikes are delta functions and the spike heights represent their areas in the sampled signal $f_\text{s}(t)$. The discrete-time signal $f[n]$ conveniently represents the same information without the annoying delta-function placeholders.

Next we find the Fourier transform of $f_\text{s}(t)$ in order to compare it to the discrete-time Fourier transform of $f[n]$. Multiplication of time representations is, by duality, equivalent to convolution of frequency representations then division by an annoying 2π. The Fourier transform of the unit comb is a frequency comb with spacing 2π and amplitude 2π. Convolving $f(\omega)$ with this comb produces a periodic transform $f_\text{s}(\omega)$. *Sampling in time produces a periodic frequency representation.*

The simplest way to find $f_\text{s}(\omega)$ is not direct convolution. It is easier to transform $f_\text{s}(t)$ directly, since it is composed of only a few simple parts (delta functions). Here is the definition of $f(t)$ again, which we need in order to find the amplitudes of the delta functions:

$$f(t) = \begin{cases} 0 & \text{for } t < -3; \\ (t+3)^2/2 & \text{for } -3 \le t \le -1; \\ 3 - t^2 & \text{for } -1 \le t \le 1; \\ (t-3)^2/2 & \text{for } 1 \le t \le 3; \\ 0 & \text{for } t > 3. \end{cases}$$

The samples are at integer times (a choice we made for simplicity). The only nonzero samples are

$$\begin{aligned} f(-2) = f(2) &= 1/2, \\ f(0) &= 3, \\ f(-1) = f(1) &= 2. \end{aligned}$$

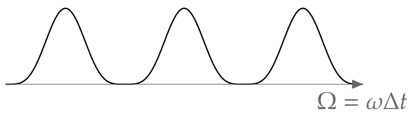
So $f_\text{s}(t)$ is the sum

$$f_\mathrm{s}(t) = \frac{1}{2}\Big(\delta(t+2) + \delta(t-2)\Big) + 3\delta(t) + 2\Big(\delta(t+1) + \delta(t-1)\Big).$$

The transform $f_\mathrm{s}(\omega)$ is

$$f_\mathrm{s}(\omega) = 3 + 4\cos\omega + \cos 2\omega,$$

which looks like



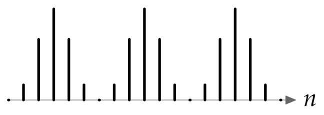
It is periodic, which it should be, and the period is 2π. In general, the period will be $2\pi/\Delta t$.

The discrete-time Fourier transform (DTFT) represents this same information slightly more conveniently by using dimensionless frequency $\Omega = \omega\Delta t$ as the independent variable instead of ω. Except for that change, the DTFT is identical to the Fourier transform of the sampled signal $f_\mathrm{s}(t)$. In this example $\Delta t = 1$, so even the Ω and ω axes are the same.

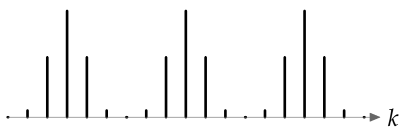
21.4 Discrete-time Fourier series

Our final Fourier representation is the discrete-time Fourier series (DTFS). This representation is often called the discrete Fourier transform (DFT) or the fast Fourier transform (FFT) in its usual algorithmic implementation. Those names misleadingly suggest that the frequency spectrum is continuous, so we will keep calling it the discrete-time Fourier series. Although this representation is the most specialized of the four, it is also the most useful because, being discrete, it can be done by a computer without needing to do difficult or impossible symbolic integrations, and because it has an efficient implementation in the so-called fast Fourier transform (that misleading term again).

The continuous-time Fourier series arises from the continuous-time Fourier transform by making the signal $f(t)$ periodic. Similarly, the discrete-time Fourier series arises from the discrete-time Fourier transform by making the sampled signal $f_\mathrm{s}(t)$ periodic. Here is the resulting periodic, sampled signal $f_\mathrm{ps}(t)$:



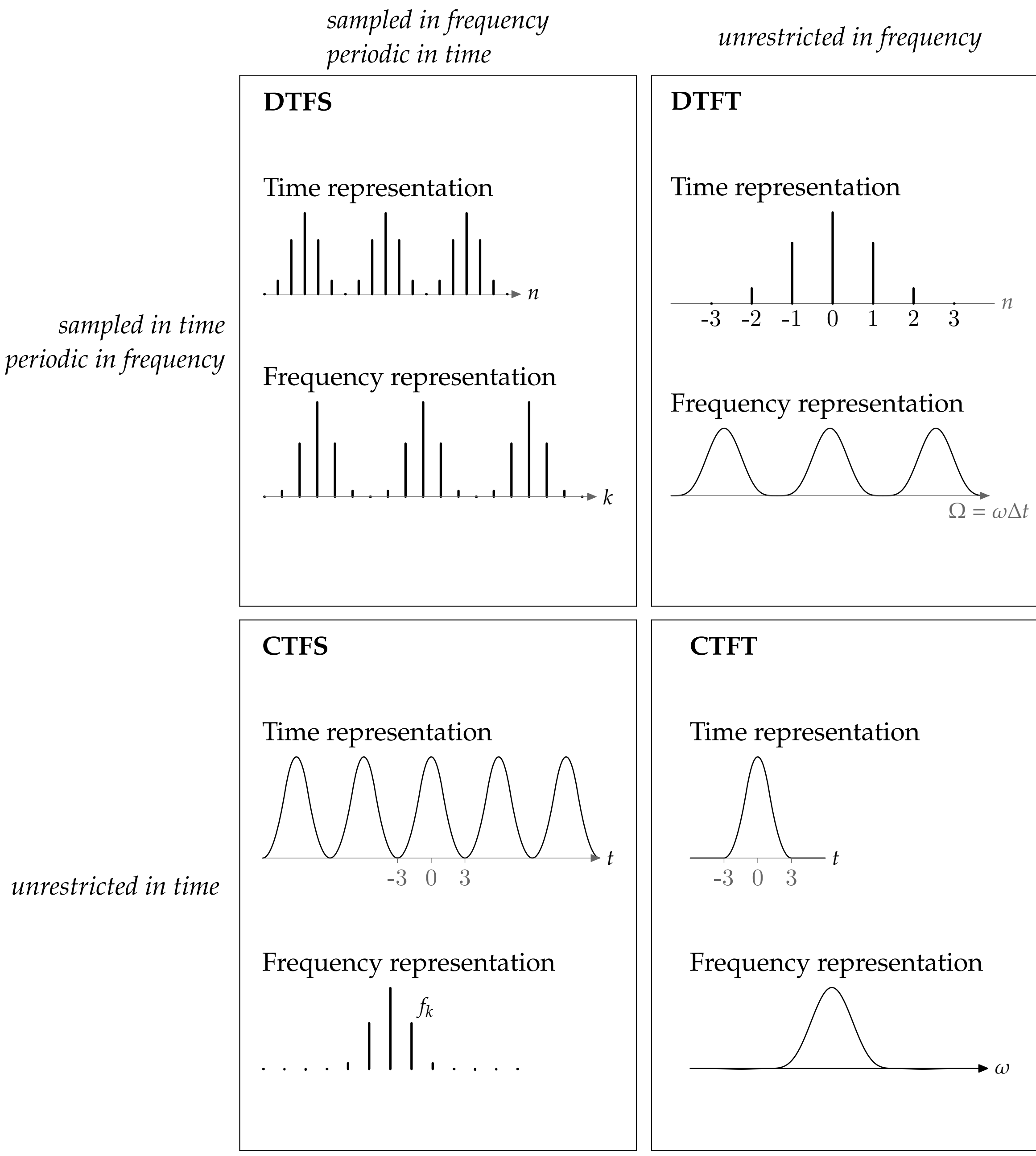
To make this signal, use the usual procedure of convolving $f_\mathrm{s}(t)$ with the comb. This comb's teeth are unit delta functions spaced by $T = 6$ time units. Convolving with the time comb is equivalent to multiplying by the frequency comb. Therefore, the Fourier transform of $f_\mathrm{ps}(t)$ is a sampled version of the discrete-time Fourier transform:



Since the discrete-time Fourier transform is periodic in frequency, the Fourier transform of $f_\mathrm{ps}(t)$ is, like $f_\mathrm{ps}(t)$ itself, both periodic and sampled. The discrete-time Fourier series conveniently represents the same information as the Fourier transform by giving the delta-function amplitudes only for one period of the transform.

21.5 Summary

This diagram shows the time signal as altered for each Fourier representation, along with the corresponding frequency representation.



Recitation 22
Sampling and reconstruction

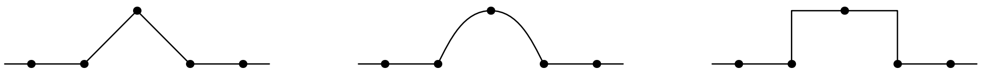
After studying this chapter, you should be able to:

- reconstruct bandlimited signals from their samples, if the samples are frequent enough; and
- explain the factor of 2 in the sampling theorem.

Unless you believe the craziness about string theory, the world is a continuous device. Yet computers are discrete devices. For a computer to represent continuous-time signals usually requires that we *sample* a signal before handing it to the computer. Sampling turns a continuous-time signal $x(t)$ and into a discrete-time signal $x[n]$ using the recipe

$$x[n] = x(nT),$$

where T is the sampling interval. In general this operation destroys information because it replaces the uncountable infinity of points in $x(t)$ with a countable infinity of points in $x[n]$. A process that destroys information must be irreversible, so it is impossible in general to reconstruct the original signal $x(t)$ from its samples. If $x[n]$ is the unit sample (the impulse), for example, here are several continuous-time signals that pass through those samples:



Requiring that $x(t)$ be continuous eliminates the third candidate but does not remove the ambiguity because the first two candidates are still viable (as are an infinity of others).

Even though continuity of $x(t)$ does not guarantee that sampling preserves information, stronger conditions guarantee that sampling does not destroy information and is therefore reversible. Finding those conditions and recovering a signal from its samples are the themes of this chapter.

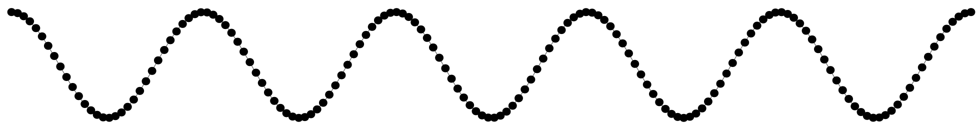
Why not forget about sampling and these difficulties and instead agree to process continuous-time signals using only analog hardware like *LRC* filters? The answer is that it is often faster and cheaper to program a computer than it is to build an analog processor. Therefore, sampling is fundamental to modern engineering.

22.1 When does sampling not destroy information?

Reconstructing $x(t)$ is hopeless if it can wiggle arbitrarily between samples. The first requirement on $x(t)$, therefore, is that it not wiggle too fast. Fast wiggles arise from high frequencies, so $x(t)$ should not contain arbitrarily high frequencies. Therefore, $x(t)$ should be *bandlimited:* Its Fourier transform should be zero for high-enough frequencies.

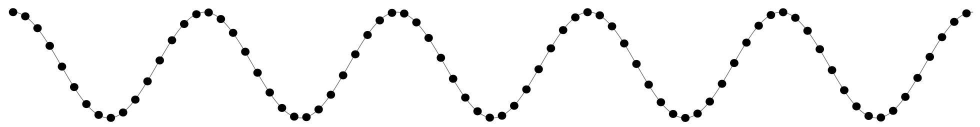
The requirement of no arbitrary wiggles is necessary but not sufficient. To find the other necessary conditions, let's sample a cosine, which is the simplest bandlimited signal. For even more simplicity, we sample $\cos t$, which is the cosine with the simplest angular frequency ($\omega = 1$).

Here are samples (black dots) from $\cos t$ using the sampling interval $T = 0.2$:

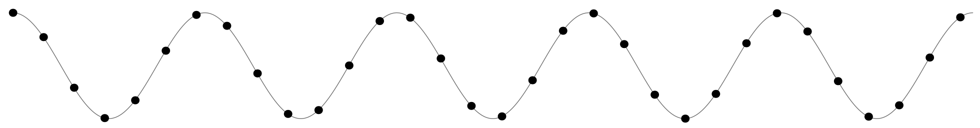


If you know that $\omega = 1$ is the highest frequency in $x(t)$, only one reconstruction is possible, namely that $x(t) = \cos t$.

Using such a tiny sampling interval is overkill for this $x(t)$. Let's successively slow down the sampling (increase the interval) until the reconstruction becomes ambiguous. Here is the same cosine sampled using $T = 0.4$:

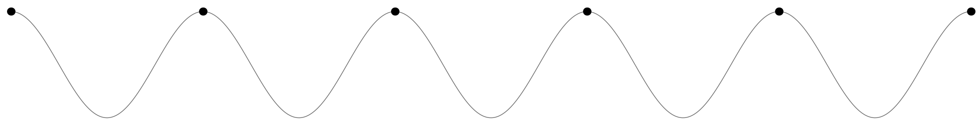


And here it is sampled using $T = 1$:



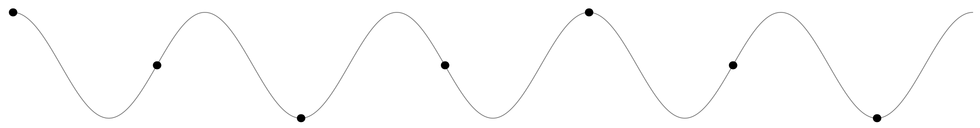
In both cases, the samples determine the reconstruction of $x(t)$, assuming again that the highest frequency is $\omega = 1$.

So even $T = 1$ is overkill. Let's therefore increase T a lot and see whether we can still reconstruct $x(t)$. Here is $T = 2\pi$:

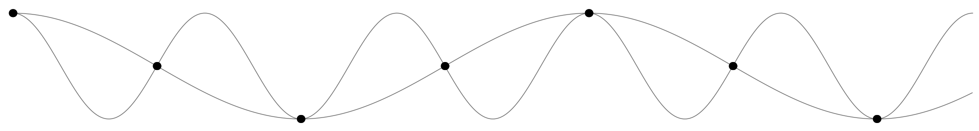


Oh no: At least two reconstructions are possible. One is $x(t) = 1$, which is a zero-frequency cosine, and another is the original signal $x(t) = \cos t$. The signal $\cos t$ has another reconstruction – an *alias* – because its frequency is too high relative to the sampling frequency. Since we overshot the critical sampling interval T, we should reduce T in order to find the critical T that avoids aliases.

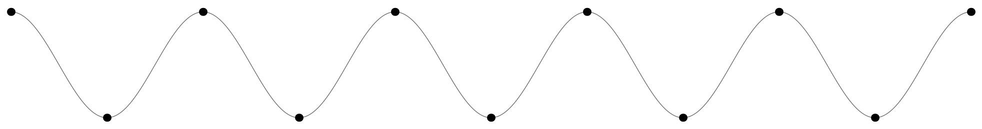
Here is the figure for a smaller sampling interval $T = 3\pi/2$:



Alas, it also has two reconstructions:



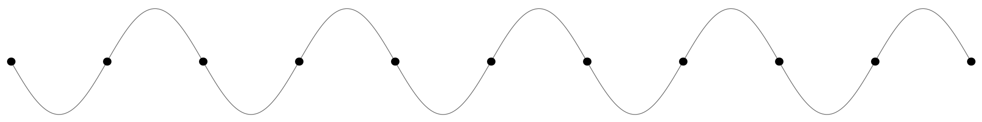
The problem in the examples with ambiguous reconstruction is that $x(t)$ wiggles between samples. When the samples are too far apart, $x(t)$ can wiggle. So the second requirement for reconstruction is frequent sampling. How frequent? The borderline case is $T = \pi$:



With this sampling interval, $x(t)$ tries to wiggle (change direction) but the next sample arrives just before $x(t)$ can do so, thereby constraining the reconstruction to be $\cos t$.

Exercise 100. Can you reconstruct any signal with frequency $\omega = 1$ using the borderline sampling interval $T = \pi$?

Sampling at exactly this borderline interval can destroy information. Try sampling the shifted cosine $\cos(t - \pi/2)$ using $T = \pi$:



The samples are all zero! To avoid this degenerate case, use $T < \pi$. Therefore the second requirement for reconstruction is to sample at more than twice the frequency of the cosine. This conclusion works for signals with more than one frequency, becoming: *To preserve information, sample at more than* ***twice*** *the highest frequency in the signal.*

This **sampling theorem** is most easily proved in the Fourier representation, which is done in Lecture 22. But the argument using the time representation gives a complementary intuition for why the theorem is true and for why it contains that otherwise mysterious factor of 2π.

22.2 Procedure for sampling and reconstruction

Sampling converts a continuous-time signal into discrete-time signal, and reconstruction converts a discrete-time signal into a continuous-time signal. Sampling followed by reconstruction therefore turns a continuous-time signal into another continuous-time signal. If sampling destroys information in the starting signal, then the starting and ending signals will not be identical. We will study the effect of sampling by comparing the starting and ending signals. Sampling itself is a simple operation, at least mathematically, so the interesting changes happen in the reconstruction operation.

Reconstruction has two conceptual steps. The first step is to turn the discrete-time sequence $x[n]$ into a continuous signal. A natural way to do so is to replace each sample with a delta function at $t = nT$ and area $x[n]$. The intermediate continuous-time signal is then

$$\sum_n x[n]\delta(t - nT) = x(t) \times \text{time comb with spacing } T.$$

The resulting signal has an infinite number of infinite discontinuities, so it contains all frequencies. Worse, their amplitudes do not fall to zero even when ω goes to infinity. In other words, this signal is as far from being bandlimited as a signal can be.

The second step in the reconstruction is to smooth this nasty signal to recover an approximation to the original signal. Smoothing means to deemphasize high frequencies. In bandlimited reconstruction, which is based on the sampling theorem, we take the extreme approach: Completely remove the high frequencies. If the sampling interval is T, then – following the procedure in **Section 22.1** for reconstructing $\cos t$ – we discard frequencies outside the range $[-\omega_s/2, \omega_s/2]$, where $\omega_s = 2\pi/T$ is the (angular) sampling frequency. The resulting low-pass-filtered signal is the reconstructed signal.

Let's compare this Fourier analysis of sampling of reconstruction to reconstructing $\cos t$ using the Force as we did in **Section 22.1**: i.e. by looking at the samples and drawing the cosine that goes through them.

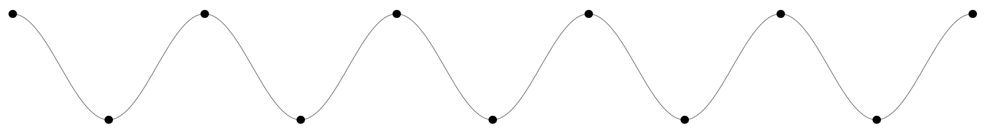
Multiplying $\cos t$ by the time comb convolves the frequency representation of $\cos t$ with the frequency representation of the time comb. The frequency representation of the time comb is a frequency comb: a train of delta functions in frequency. The starting signal $\cos t$ has frequency representation

$$x(\omega) = \pi\delta(t - 1) + \pi\delta(t + 1),$$

Convolving $x(\omega)$ with a frequency comb copies the two delta functions over and over at ever higher frequencies. The important conclusion of all this comb gymnastics is that the sampled continuous-time signal has a periodic frequency representation. This consequence is familiar from **Section 21.3** where we compared the discrete-time Fourier transform with the continuous-time Fourier transform of a sampled signal.

In the second step of reconstruction, we low-pass filter this sampled continuous-time signal by discarding all but the first period of its frequency representation. This operation leaves us the original signal $x(t) = \cos t$, as long as the sampling interval is small enough. How small is small enough? We have an answer from the sampling theorem, but let's find the same answer using this frequency-representation analysis. To do so, imagine increasing the sampling interval until a problem appears. As the sampling interval T is increases, the frequency comb's teeth move closer. Convolving with a frequency comb causes a problem when its tooth spacing is smaller than the width of $x(\omega)$, where the width means the width of the nonzero region of $x(\omega)$. When that happens, the copies overlap, which is the frequency representation of aliasing, which destroys information about the original signal.

For $x(t) = \cos t$, the maximum frequency is $\omega_{\max} = 1$ so the width is 2 because $x(t)$ (like any real signal) contains positive and negative frequencies. Since the tooth spacing for the frequency comb is $2\pi/T$, the condition for accurate reconstruction (equivalently, for no aliasing) is that $2\pi/T > 2$ or $T < \pi$, which is the critical sampling interval that we found in **Section 22.1** and is illustrated again in this figure



22.3 Impulse response of reconstruction

A cosine is a simple and therefore useful signal with which to understand sampling and reconstruction. It is the extreme case of a signal with the most peaked Fourier transform. The other extreme is the function with the flattest Fourier transform, which is the most peaked time signal $x(t) = \delta(t)$. Rather than look directly at sampling and reconstructing $\delta(t)$, we'll look at a related but more useful case of reconstructing the most peaked discrete-time signal. We are skipping over the sampling operation and starting with $x[n]$ because sampling – turning the continuous-time signal $x(t)$ into $x[n]$ – is such a trivial operation mathematically; engineering it is very hard, but that problem is another story. The only mathematical choice in sampling is the sampling interval T. The mathematically interesting operation is reconstruction, and many methods are possible. This chapter focuses on bandlimited reconstruction, so let's investigate that method.

Pause to try 80. Is sampling followed by bandlimited reconstruction a linear operation?

Sampling – turning $x(t)$ into $x[n]$ – is a linear operation. What about reconstruction? Bandlimited reconstruction means multiplying the frequency representation by a pulse to throw out the high frequency copies. Reconstruction therefore convolves the sampled function with the time representation of a frequency pulse. Convolution is a linear operation, so reconstruction is also a linear operation. Therefore, the combination of sampling followed by bandlimited reconstruction is also a linear operation! The exclamation mark is because the combined operation uses multiplication in time and in frequency, and one of those multiplications could have made the whole operation nonlinear. But each multiplication is by a function independent of the signal, so the operations are still linear.

Pause to try 81. Is sampling followed by bandlimited reconstruction a time-invariant operation?

As long as the sampling does not destroy information – meaning that no frequencies alias – then sampling followed by bandlimited reconstruction exactly reconstructs the starting signal, whether or not it is shifted in time. So the combined operation of sampling and reconstruction is time invariant if the signal is properly bandlimited. In general, however, the combined operation is not time invariant: For example, reconstructing the shifted cosine in **Section 22.1** using $T = \pi$ produces $x(t) = 0$ because sampling at slightly too low a rate destroyed information.

To probe bandlimited reconstruction, we find how it reconstructs the most peaked discrete-time signal, which is the unit sample $\delta[n]$ (also known as the discrete-time impulse). We are therefore computing the impulse response of the reconstruction operation. An impulse has a flat Fourier transform. Low-pass filtering it by chopping off all frequencies beyond a cutoff frequency convolves the time representation with a sinc function, where

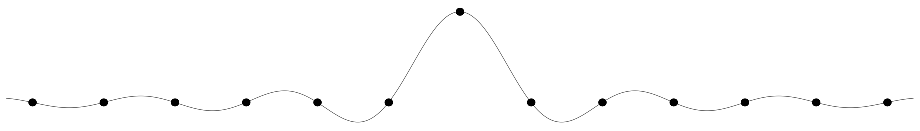
$$\operatorname{sinc} x \equiv \frac{\sin x}{x}.$$

To avoid mistakes with 2π and T, we'll guess the amplitude and t-axis scaling of the sinc rather than compute it directly. A reasonable requirement on the reconstructed signal $x_{\mathrm{r}}(t)$ is that it at least pass through the samples $x(nT)$. So the reconstruction of $\delta[n]$ should be 1 at $t = 0$ and 0 at

$t = nT$ for $n \neq 0$. A properly scaled sinc function such as $\alpha \operatorname{sinc} \beta t$ can meet both requirements. Since $\operatorname{sinc} 0 = 1$, the requirement that $x_r(0) = 1$ means $\alpha = 1$. The zeros of $\operatorname{sinc} \beta t$ are the zeros of $\sin \beta t$ (except at $t = 0$). Those zeros are $t = n\pi/\beta$ for $n \neq 0$. To put the zero crossings of the sinc at nT requires $\pi/\beta = T$ or $\beta = \pi/T$. So the reconstruction of the unit sample is

$$x_r(t) = \operatorname{sinc} \frac{\pi t}{T},$$

which looks like



The unit-sample response is useful because it gives a reconstruction procedure starting with a general discrete-time signal.

Exercise 101. Here is a continuous-time signal: $x(t)$ is $1 - |t|$ for $|t| \leq 1$ and is zero elsewhere. Sample it with $T = 1$ and obtain its reconstruction $x_r(t)$. Sketch $x(t)$ and $x_r(t)$ and mark the samples.

Exercise 102. Here is another continuous-time signal: $x(t)$ is $1 - t^2$ for $|t| \leq 1$ and is zero elsewhere. Sample it with $T = 1$ and obtain its reconstruction $x_r(t)$. Sketch $x(t)$ and $x_r(t)$ and mark the samples.

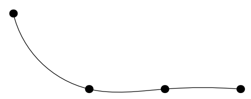
Exercise 103. Compare the two preceding answers and explain the comparison in terms of frequency representations.

Exercise 104. Here is a more interesting continuous-time signal: $x(t)$ is 1 for $|t| \leq 3/2$ and is zero elsewhere. Sample it with $T = 1$ and obtain its reconstruction $x_r(t)$. Sketch $x(t)$ and $x_r(t)$ and mark the samples.

Recitation 23
Interpolation as reconstruction

After studying this chapter, you should be able to:

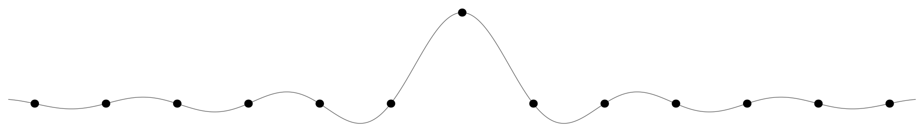
- explain bandlimited, piece constant, and piecewise linear reconstruction using either the time or frequency representations; and
- write a program that uses interpolation to compute a mathematical function such e^t or $\sin t$.



Reconstruction takes a signal defined at only particular times – a discrete signal – and produces a signal defined for all intermediate times. The reconstructed signal *interpolates* between the samples of the discrete signal. The figure shows one interpolation that passes through the given samples, although an infinity of interpolations are possible. Each method of reconstruction has advantages and disadvantages. In general, simpler methods produce interpolations that are less smooth. In this chapter we try out a few methods for reconstructing e^t. The exponential function is essential in mathematics and physics, and our reconstructions provide methods for a computer to evaluate e^t, whether in software such as a mathematics library function or in hardware such as a floating-point processor.

23.1 Bandlimited reconstruction

We already know bandlimited reconstruction (**Section 22.2**). Bandlimited reconstruction has a simple picture in the frequency representation: Smooth the sampled signal by completely discarding its high frequencies. Like almost every reconstruction method, this operation acts as a low-pass filter. Discarding high frequencies means multiplying the frequency representation by a pulse. Its effect is analyzed in **Section 22.3**, which computes its impulse response. The response is $\operatorname{sinc}\frac{\pi t}{T}$, where T is the sampling interval, and looks like



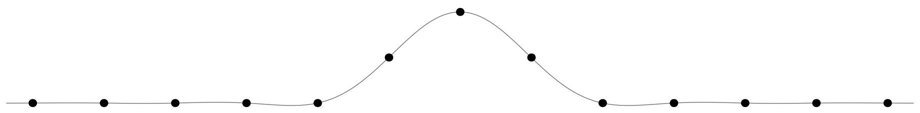
That picture is the bandlimited reconstruction of the unit sample. Since any discrete-time signal is a sum of shifted unit samples, we can reconstruct a signal from its discrete-time samples by adding shifted sincs. Suppose that the original signal is the triangle $x(t) = 2 - |t|$ for $t = -2 \ldots 2$ (and zero otherwise) and it is sampled with $T = 1$ to get

$$x[n] = \begin{cases} 2 - |n| & \text{for } n = -1 \ldots 1; \\ 0 & \text{otherwise.} \end{cases}$$

Each nonzero $x[n]$ contributes a shifted sinc to the reconstruction:

$$x_r(t) = \underbrace{1 \times \operatorname{sinc}\frac{\pi(t+T)}{T}}_{\text{from } x[-1]=1} + \underbrace{2 \times \operatorname{sinc}\frac{\pi(t)}{T}}_{\text{from } x[0]=2} + \underbrace{1 \times \operatorname{sinc}\frac{\pi(t-T)}{T}}_{\text{from } x[1]=1}$$

which looks like



The impulse response is infinitely smooth, meaning that every derivative is continuous. Therefore bandlimited reconstruction, which adds up these infinitely smooth responses, produces an infinitely smooth reconstruction. As an example of that smoothing, the original triangle $x(t)$ has a slope discontinuity at $t = 0$ but the its reconstruction has a smooth bump at $t = 0$.

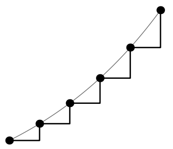
Smooth reconstruction is an advantage of bandlimited reconstruction, as long as one is trying to reconstruct infinitely smooth signals. This restriction is reasonable because no physical signal has discontinuities (unless you believe the string-theory craziness). Another advantage of bandlimited reconstruction is its simple frequency picture.

Despite this theoretical cleanliness, the method has the large disadvantage that its impulse response extends forever. Therefore to reconstruct a signal anywhere, you need to know *all* of its samples or, more precisely, all of its nonzero samples. The triangle example had only three nonzero samples, making it easy to write a closed form for the reconstruction and to implement it numerically. However, what if you want to reconstruct a realistic function such as e^t in the range $t = 0\ldots1$? The numerical routine would need to store samples of the exponential function for all time. Not only would these samples require an infinite amount of money to buy memory or disk space, the reconstruction would not even converge because the samples grow so rapidly for large positive t.

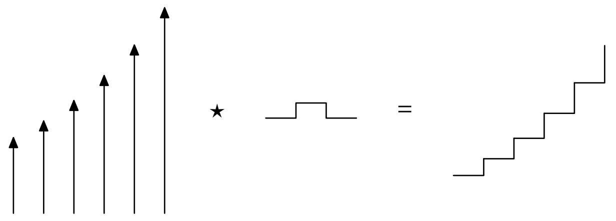
23.2 Piecewise constant reconstruction

The simplest alternative to bandlimited reconstruction is piecewise constant reconstruction. It is also called zeroth-order hold because it holds (preserves) the previous sample until the next sample arrives. Let's see how well it interpolates e^t in the range $t = 0\ldots1$.

Exercise 105. Given a function that computes e^t for $t = 0\ldots1$, how would you compute e^t for any t?



The margin figure shows e^t for $t = 0\ldots1$ sampled with interval $T = 0.2$ and reconstructed using piecewise constant reconstruction. The smooth curve is e^t. Although piecewise constant reconstruction. produces an awful rendition of e^t, it is simple to describe in the time representation: Convolve the sampled continuous-time signal (a sum of delta functions) with a pulse of height 1 and width T, which is the unit-sample response of piecewise constant reconstruction. The following picture illustrates the convolution:



The method is simple to implement, an advantage especially if it will be implemented in hardware rather than in software. However, its simplicity has a cost, which you can figure out by doing the next problem.

Pause to try 82. If the implementation is supposed to calculate e^t in the range $t = 0 \ldots 1$ with an accuracy of 10^{-6} (six digits), roughly how many samples do you need?

The maximum error is the size of jump at $t = 1$, which is approximately

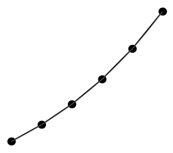
sampling interval $\times$ slope of e^t at $t = 1$.

Since the slope at $t = 1$ is of order 1, the jump is roughly the sampling interval T. So six-digit accuracy requires $T \sim 10^{-6}$, which means storing 10^6 samples! Piecewise constant interpolation needs a a lot of storage to get reasonable accuracy.

Exercise 106. What does piecewise constant reconstruction look like in the frequency representation? Is its impulse response bandlimited?

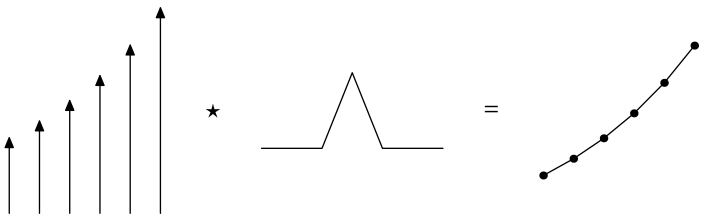
23.3 Piecewise linear reconstruction

An improved method is to connect the dots with straight lines. This method is piecewise linear reconstruction. Alternatively it is called first-order hold because, like zeroth-order hold, it holds the previous sample until the next arrives but unlike zeroth-order hold, it connects them continuously. The margin figure shows this reconstruction of e^t alongside e^t itself. It is difficult to distinguish the two curves.



Pause to try 83. Piece constant reconstruction convolves the sampled signal with a pulse. What is the corresponding description for piecewise constant reconstruction?

Making the connection continuous requires knowing the future. After a sample arrives, you have to decide whether to go upward or downward, a decision that requires knowing the next sample. A sample therefore affects the reconstruction for a time T before it arrives and for a time T after it arrives, making the unit-sample response of this reconstruction operation noncausal. The unit-sample response of this method is a triangle of width $2T$ and unit height, as you can check by doing the following convolution:



Piecewise linear reconstruction is an improvement on piecewise constant reconstruction. It requires far fewer samples than piecewise constant reconstruction, and its implementation is almost as sample.

Pause to try 84. How many samples do you need for six digits accuracy?

In piecewise constant reconstruction, the error was proportional to the sampling interval T. Piecewise linear reconstruction takes account of linear behavior between samples, but it does not notice quadratic effects. So the error will be proportional to T^2. Six-digit accuracy means an error of 10^{-6}, so we need $T \sim 10^{-3}$ or about 1000 samples. Compared to piecewise constant reconstruction, piecewise linear interpolation reduces the storage by a factor of 1000. The improvement is even more drastic when interpolating at higher accuracy. For example, 16-digit accuracy – which is roughly the maximum accuracy with double-precision floating-point numbers – would require 10^{16} samples using piecewise constant reconstruction but only 10^8 samples using piecewise linear reconstruction. It is still a space monster but less so by a factor of 10^8.

Exercise 107. What does piecewise linear reconstruction look like in the frequency representation? Compare it to piecewise constant reconstruction.

Exercise 108. Write a program that uses samples of e^t to do piecewise linear interpolation.

23.4 Smoother reconstructions

Just as piecewise linear is more accurate than piecewise constant reconstruction, piecewise quadratic reconstruction is more accurate than piecewise linear reconstruction. Piecewise quadratic reconstruction interpolates (reconstructs) at $t = t_0$ by fitting a parabola through the three samples nearest to t_0.

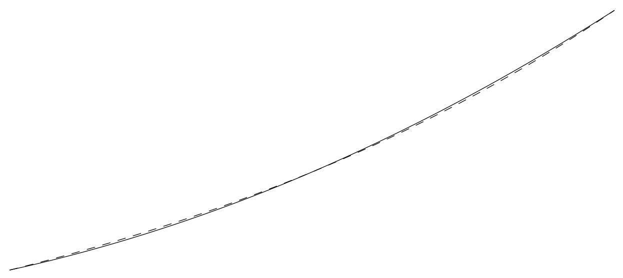
The crudest quadratic approximation to e^t in the range $0 \ldots 1$ is when $T = 0.5$. A larger interval produces too few samples to determine the quadratic. With $T = 0.5$ the samples are

$$(0, 1), \quad (0.5, \sqrt{e}), \quad (1, e),$$

and the parabola passing through them is

$$y(t) \approx 1.00000 + 0.87660 * t + 0.84168 * t * t.$$

[The `polyfit` function in Octave/Matlab will find the coefficients.] Here is a graph of the fitted parabola (the solid line) alongside e^t (the dashed line):



Even though we used only three samples, the approximation is hard to distinguish from the exact curve.

Pause to try 85. What is the biggest error in this simple quadratic fit?

The biggest error happens when the difference between $y(t)$ and e^t is a maximum or a minimum, i.e. when it has zero slope. The resulting equation for the derivative is transcendental, but the Newton–Raphson method finds its solutions. The `maxima` symbolic-mathematics program has a Newton–Raphson routine, and it says that the maximum error happens at $t \approx 0.796$ when the error is approximately 0.014. A sampling interval of $T = 0.5$ provides almost two decimal places accuracy!

Pause to try 86. Roughly how many samples are needed for six digits of accuracy?

Piecewise linear interpolation has an error proportional to T^2, so quadratic interpolation should have an error proportional to T^3. To get four more digits of accuracy, which is a factor of 10^4, we should reduce the sampling interval by the cube root of 10^4. The cube root is roughly 22 so a sampling interval of around 0.02 should give six-digit accuracy.

To test that prediction, I wrote a short Python program to do polynomial interpolation. You'll do so on the next problem set, so I'll just quote its result: With $T = 0.02$, the quadratic interpolator has a worst-case error of 1.4×10^{-6}, meaning that we get six-digit accuracy with the predicted sampling interval $T = 0.02$.

Exercise 109. Write a program that uses samples of e^t to do quadratic interpolation (in the usual range $t = 0 \ldots 1$).

Exercise 110. Write a program that interpolates e^t to 15-digit accuracy in our usual range.

Bibliography

[1] Central Intelligence Agency. *The World Factbook*. Central Intelligence Agency, 2007. https://www.cia.gov/library/publications/the-world-factbook/.

[2] Richard Feynman and Ralph Leighton (contributor). *Surely You're Joking, Mr. Feynman! Adventures of a Curious Character*. W. W. Norton, 1985.

[3] Leah Edelstein-Keshet. *Mathematical models in biology*. SIAM, Philadelphia, 2005.

[4] Fibonacci. *Liber Abaci.*, 1202.

[5] Albert Einstein. Zur elektrodynamik bewegter körper [On the electrodynamics of moving bodies]. *Annalen der Physik*, 17:891–921, 1905.

[6] Hermann Minkowski H. A. Lorentz Albert Einstein and Hermann Weyl. *The Principle of Relativity: A Collection of Original Memoirs*. Dover, 1952.

[7] *Spacetime Physics*. W. H. Freeman and Co., 1992.

[8] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[9] Tom R. Halfhill. An error in a lookup table created the infamous bug in Intel's latest processor. *BYTE*, March 1995. http://www.byte.com/art/9503/sec13/art1.htm.

[10] P. Ribenboim. *The New Book of Prime Number Records*. Springer–Verlag, New York, 1996.

[11] Jonathan Borwein and David Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st century*. A K Peters, 2003.

[12] David Bailey Jonathan Borwein and Roland Girgensohn. *Experimentation in Mathematics: Computational Paths to Discovery*. A K Peters, 2004.

[13] David Epstein and Sylvio Levy. Experimentation and proof in mathematics. *Notices of the American Mathematical Society*, pages 670–674, June/July 1995.

[14] Jan Brett (illustrator). *Goldilocks and the Three Bears*. Dodd, 1987.

[15] Tjalling J. Ypma. Historical development of the Newton–Raphson method. *SIAM Review*, 37(4):531-551,, 1995.

Index