Discrete-time Signals and Systems

Discrete-time Signals and Systems

An Operator Approach

Sanjoy Mahajan and Dennis Freeman
Massachusetts Institute of Technology

Typeset in Palatino and Euler by the authors using ConTEXt and PDFTEX

Source revision: 66261db0f9ed+ (2009-10-18 13:33:48 UTC)

Brief contents

	Preface	ix
1	Difference equations	1
2	Difference equations and modularity	17
3	Block diagrams and operators: Two new representations	33
4	Modes	51
5	Repeated roots	63
6	The perfect (sine) wave	71
7	Control	83
8	Proportional and derivative control	95
	Bibliography	105
	Index	107

Contents

Preface **ix**

1 Difference equations **1**
1.1 Rabbits 2
1.2 Leaky tank 7
1.3 Fall of a fog droplet 11
1.4 Springs 14

2 Difference equations and modularity **17**
2.1 Modularity: Making the input like the output 17
2.2 Endowment gift 21
2.3 Rabbits 25

3 Block diagrams and operators: Two new representations **33**
3.1 Disadvantages of difference equations 34
3.2 Block diagrams to the rescue 35
3.3 The power of abstraction 40
3.4 Operations on whole signals 41
3.5 Feedback connections 45
3.6 Summary 49

4 Modes **51**
4.1 Growth of the Fibonacci series 52
4.2 Taking out the big part from Fibonacci 55
4.3 Operator interpretation 57
4.4 General method: Partial fractions 59

5 Repeated roots **63**
5.1 Leaky-tank background 64
5.2 Numerical computation 65
5.3 Analyzing the output signal 67

5.4 Deforming the system: The continuity argument 68
5.5 Higher-order cascades 70

6 The perfect (sine) wave 71
6.1 Forward Euler 72
6.2 Backward Euler 76
6.3 Leapfrog 79
6.4 Summary 82

7 Control 83
7.1 Motor model with feedforward control 83
7.2 Simple feedback control 85
7.3 Sensor delays 87
7.4 Inertia 90

8 Proportional and derivative control 95
8.1 Why derivative control 95
8.2 Mixing the two methods of control 96
8.3 Optimizing the combination 98
8.4 Handling inertia 99
8.5 Summary 103

Bibliography 105

Index 107

Preface

This book aims to introduce you to a powerful tool for analyzing and designing systems – whether electronic, mechanical, or thermal.

This book grew out of the 'Signals and Systems' course (numbered 6.003) that we have taught on and off to MIT's Electrical Engineering and Computer Science students.

The traditional signals-and-systems course – for example [17] – emphasizes the analysis of continuous-time systems, in particular analog circuits. However, most engineers will not specialize in analog circuits. Rather, digital technology offers such vast computing power that analogy circuits are often designed through digital simulation.

Digital simulation is an inherently discrete-time operation. Furthermore, almost all fundamental ideas of signals and systems can be taught using discrete-time systems. Modularity and multiple representations , for example, aid the design of discrete-time (or continuous-time) systems. Similarly, the ideas for modes, poles, control, and feedback.

Furthermore, by teaching the material in a context not limited to circuits, we emphasize the generality of these tools. Feedback and simulation abound in the natural and engineered world, and we would like our students to be flexible and creative in understanding and designing these systems.

Therefore, we begin our 'Signals and Systems' course with discrete-time systems, and give our students this book. A fundamental difference from most discussions of discrete-time systems is the approach using operators. Operators make it possible to avoid the confusing notion of 'transform'. Instead, the operator expression for a discrete-time system, and the system's impulse response are two representations for the same system; they are the coordinates of a point as seen from two different coordinate systems. Then a transformation of a system has an active meaning: for example, composing two systems to build a new system.

How to use this book

Aristotle was tutor to the young Alexander of Macedon (later, the Great). As ancient royalty knew, a skilled and knowledgeable tutor is the most effective teacher [3]. A skilled tutor makes few statements and asks many questions, for she knows that questioning, wondering, and discussing promote long-lasting learning. Therefore, questions of two types are interspersed through the book:

questions marked with a ▶ in the margin: These questions are what a tutor might ask you during a tutorial, and ask you to work out the next steps in an analysis. They are answered in the subsequent text, where you can check your solutions and my analysis.

numbered questions: These problems, marked with a shaded background, are what a tutor might ask you to take home after a tutorial. They give further practice with the tool or ask you to extend an example, use several tools together, or resolve paradoxes.

Try lots of questions of both types!

Copyright license

Acknowledgments

We gratefully thank the following individuals and organizations:

For suggestions and discussions: . . .

For the free software for typesetting: Donald Knuth (TEX); Han The Thanh (PDFTEX); Hans Hagen and Taco Hoekwater (ConTEXt); John Hobby (MetaPost); Andy Hammerlindl, John Bowman, and Tom Prince (`asymptote`); Richard Stallman (`emacs`); the Linux and Debian projects.

1
Difference equations

1.1 Rabbits 2
1.2 Leaky tank 7
1.3 Fall of a fog droplet 11
1.4 Springs 14

The world is too rich and complex for our minds to grasp it whole, for our minds are but a small part of the richness of the world. To cope with the complexity, we reason hierarchically. We divide the world into small, comprehensible pieces: systems. Systems are ubiquitous: a CPU, a memory chips, a motor, a web server, a jumbo jet, the solar system, the telephone system, or a circulatory system. Systems are a useful abstraction, chosen because their external interactions are weaker than their internal interactions. That properties makes independent analysis meaningful.

Systems interact with other systems via forces, messages, or in general via information or signals. 'Signals and systems' is the study of systems and their interaction.

This book studies only discrete-time systems, where time jumps rather than changes continuously. This restriction is not as severe as its seems. First, digital computers are, by design, discrete-time devices, so discrete-time signals and systems includes digital computers. Second, almost all the important ideas in discrete-time systems apply equally to continuous-time systems.

Alas, even discrete-time systems are too diverse for one method of analysis. Therefore even the abstraction of systems needs subdivision. The particular class of so-called linear and time-invariant systems admits powerful tools of analysis and design. The benefit of restricting ourselves to such

systems, and the meaning of the restrictions, will become clear in subsequent chapters.

1.1 Rabbits

Here is Fibonacci's problem [6, 10], a famous discrete-time, linear, time-invariant system and signal:

> 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 begets a new pair which from the second month on becomes productive?

1.1.1 Mathematical representation

This system consists of the rabbit pairs and the rules of rabbit reproduction. The signal is the sequence f where $f[n]$ is the number of rabbit pairs at month n (the problem asks about $n = 12$).

▶ *What is f in the first few months?*

In month 0, one rabbit pair immigrates into the system: $f[0] = 1$. Let's assume that the immigrants are children. Then they cannot have their own children in month 1 – they are too young – so $f[1] = 1$. But this pair is an adult pair, so in month 2 the pair has children, making $f[2] = 2$.

Finding $f[3]$ requires considering the adult and child pairs separately (hierarchical reasoning), because each type behaves according to its own reproduction rule. The child pair from month 2 grows into adulthood in month 3, and the adult pair from month 2 begets a child pair. So in $f[3] = 3$: two adult and one child pair.

The two adult pairs contribute two child pairs in month 4, and the one child pair grows up, contributing an adult pair. So month 4 has five pairs: two child and three adult pairs. To formalize this reasoning process, define two intermediate signals c and a:

$$c[n] = \text{number of child pairs at month } n;$$
$$a[n] = \text{number of adult pairs at month } n.$$

The total number of pairs at month n is $f[n] = c[n] + a[n]$. Here is a table showing the three signals c, a, and f:

n	0	1	2	3
c	1	0	1	1
a	0	1	1	2
f	1	1	2	3

The arrows in the table show how new entries are constructed. An upward diagonal arrow represents the only means to make new children, namely from last month's adults:

$$a[n-1] \to c[n] \qquad \text{or} \qquad c[n] = a[n-1].$$

A horizontal arrow represents one contribution to this month's adults, that adults last month remain adults: $a[n-1] \to a[n]$. A downward diagonal arrow represents the other contribution to this month's adults, that last month's children grow up into adults: $c[n-1] \to a[n]$. The sum of the two contributions is

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

▶ *What is the difference equation for f itself?*

To find the equation for f, one has at least two methods: logical deduction (Problem 1.1) or trial and error. Trial and error is better suited for finding results, and logical deduction is better suited for verifying them. Therefore, using trial and error, look for a pattern among addition samples of f:

n	0	1	2	3	4	5	6
c	1	0	1	1	2	3	5
a	0	1	1	2	3	5	8
f	1	1	2	3	5	8	13

▶ *What useful patterns live in these data?*

One prominent pattern is that the signals c, a, and f look like shifted versions of each other:

$$a[n] = f[n-1];$$
$$c[n] = a[n-1] = f[n-2].$$

Since $f[n] = a[n] + c[n]$,

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

with initial conditions $f[0] = f[1] = 1$.

This mathematical description, or representation, clarifies a point that is not obvious in the verbal description: that the number of rabbit pairs in any month depends on the number in the two preceding months. This difference equation is said to be a second-order difference equation. Since its coefficients are all unity, and the signs are positive, it is the simplest second-order difference equation. Yet its behavior is rich and complex.

Problem 1.1 Verifying the conjecture
Use the two intermediate equations

$$c[n] = a[n-1],$$
$$a[n] = a[n-1] + c[n-1];$$

and the definition $f[n] = a[n] + c[n]$ to confirm the conjecture

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

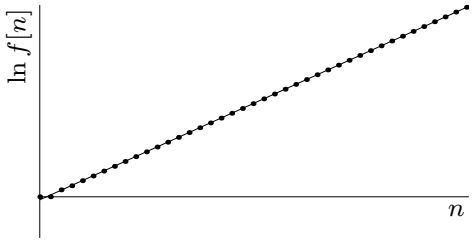
1.1.2 Closed-form solution

The rabbit difference equation is an implicit recipe that computes new values from old values. But does it have a **closed form**: an explicit formula for $f[n]$ that depends on n but not on preceding samples? As a step toward finding a closed form, let's investigate how $f[n]$ behaves as n becomes large.

▶ *Does $f[n]$ grow like a polynomial in n, like a logarithmic function of n, or like an exponential function of n?*

Deciding among these options requires more data. Here are many values of $f[n]$ (starting with month 0):

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, \ldots$$



The samples grow quickly. Their growth is too rapid to be logarithmic, unless $f[n]$ is an unusual function like $(\log n)^{20}$. Their growth is probably also too rapid for $f[n]$ to be a polynomial in n, unless $f[n]$ is a high-degree polynomial. A likely alternative is exponential growth. To test that hypothesis, use pictorial reasoning by plotting $\ln f[n]$ versus n. The plotted points oscillate above and below a best-fit straight line. Therefore $\ln f[n]$ grows almost exactly linearly with n, and $f[n]$ is approximately an exponential function of n:

$$f[n] \approx Az^n,$$

where z and A are constants.

▶ *How can z be estimated from $f[n]$ data?*

n	$f[n]/f[n-1]$
10	1.6181818181818
20	1.6180339985218
30	1.6180339887505
40	1.6180339887499
50	1.6180339887499

Because the plotted points fall ever closer to the best-fit line as n grows, the exponential approximation $f[n] \approx Az^n$ becomes more exact as n grows. If the approximation were exact, then $f[n]/f[n-1]$ would always equal z, so $f[n]/f[n-1]$ becomes an ever closer approximation to z as n increases. These ratios seem to converge to $z = 1.6180339887499$.
Its first few digits 1.618 might be familiar. For a memory amplifier, feed the ratio to the online Inverse Symbolic Calculator (ISC). Given a number, it guesses its mathematical source. When given the Fibonacci z, the Inverse Symbolic Calculator suggests two equivalent forms: that z is a root of $1 - x - x^2$ or that it is $\phi \equiv (1 + \sqrt{5})/2$. The constant ϕ is the famous golden ratio [5].

n	$f[n]/f[n-1]$
10	0.72362506926472
20	0.72360679895785
30	0.72360679775006
40	0.72360679774998
50	0.72360679774998

Therefore, $f[n] \approx A\phi^n$. To find the constant of proportionality A, take out the big part by dividing $f[n]$ by ϕ^n. These ratios hover around $0.723\ldots$, so perhaps A is $\sqrt{3} - 1$. Alas, experiments with larger values of n strongly suggest that the digits continue $0.723606\ldots$ whereas $\sqrt{3} - 1 = 0.73205\ldots$. A bit of experimentation or the Inverse Symbolic Calculator suggests that 0.72360679774998 probably originates from $\phi/\sqrt{5}$.

This conjecture has the merit of reusing the $\sqrt{5}$ already contained in the definition of ϕ, so it does not introduce a new arbitrary number. With that conjecture for A, the approximation for $f[n]$ becomes

$$f[n] \approx \frac{\phi^{n+1}}{\sqrt{5}}.$$

▶ *How accurate is this approximation?*

To test the approximation, take out the big part by computing the residual:

$$r[n] = f[n] - \phi^{n+1}/\sqrt{5}.$$

The residual decays rapidly, perhaps exponentially. Then r has the general form

$$r[n] \approx By^n,$$

where y and B are constants. To find y, compute the ratios $r[n]/r[n-1]$. They converge to $-0.61803\ldots$, which is almost exactly $1-\phi$ or $-1/\phi$. Therefore $r[n] \approx B(-1/\phi)^n$.

n	$r[n]/r[n-1]$
2	−0.61803398874989601
3	−0.61803398874989812
4	−0.61803398874988624
5	−0.61803398874993953
6	−0.61803398874974236
7	−0.61803398875029414
8	−0.61803398874847626
9	−0.61803398875421256
10	−0.61803398873859083

▶ *What is the constant of proportionality* B*?*

To compute B, divide $r[n]$ by $(-1/\phi)^n$. These values, if n is not too large (Problem 1.2), almost instantly settles on 0.27639320225. With luck, this number can be explained using ϕ and $\sqrt{5}$. A few numerical experiments suggest the conjecture

$$B = \frac{1}{\sqrt{5}} \times \frac{1}{\phi}.$$

The residual becomes

$$r[n] = -\frac{1}{\sqrt{5}} \times \left(-\frac{1}{\phi}\right)^{n+1}.$$

The number of rabbit pairs is the sum of the approximation Az^n and the residual By^n:

$$f[n] = \frac{1}{\sqrt{5}}\left(\phi^{n+1} - (-\phi)^{-(n+1)}\right).$$

How bizarre! The Fibonacci signal f splits into two signals in at least two ways. First, it is the sum of the adult-pairs signal a and the child-pairs signal c. Second, it is the sum $f_1 + f_2$ where f_1 and f_2 are defined by

$$f_1[n] \equiv \frac{1}{\sqrt{5}}\phi^{n+1};$$
$$f_2[n] \equiv -\frac{1}{\sqrt{5}}(-1/\phi)^{n+1}.$$

The equivalence of these decompositions would have been difficult to predict. Instead, many experiments and guesses were needed to establish the equivalence. Another kind of question, also hard to answer, arises by changing merely the plus sign in the Fibonacci difference equation into a minus sign:

$$g[n] = g[n-1] - g[n-2].$$

With corresponding initial conditions, namely $g[0] = g[1] = 1$, the signal g runs as follows (staring with $n = 0$):

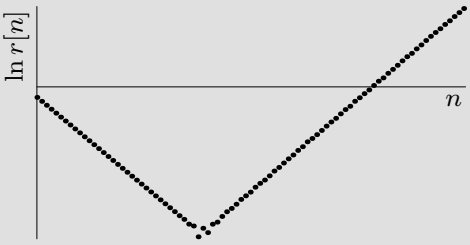
$$\underbrace{1, 1, 0, -1, -1, 0, 1,}_{\text{one period}} 1, 0, -1, -1, 0, \ldots.$$

Rather than growing approximately exponentially, this sequence is exactly periodic. Why? Furthermore, it has period 6. Why? How can this period be predicted without simulation?

A representation suited for such questions is introduced in ??. For now, let's continue investigating difference equations to represent systems.

Problem 1.2 Actual residual

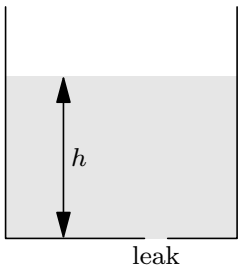
Here is a semilog graph of the absolute residual $|r[n]|$ computed numerically up to $n = 80$. (The absolute residual is used because the residual is often negative and would have a complex logarithm.) It follows the predicted exponential decay for a while, but then misbehaves. Why?



1.2 Leaky tank

In the Fibonacci system, the rabbits changed their behavior – grew up or had children – only once a month. The Fibonacci system is a discrete-time

system. These systems are directly suitable for computational simulation and analysis because digital computers themselves act like discrete-time systems. However, many systems in the world – such as piano strings, earthquakes, microphones, or eardrums – are naturally described as continuous-time systems.



To analyze continuous-time systems using discrete-time tools requires approximations. These approximations are illustrated in the simplest interesting continuous-time system: a leaky tank. Imagine a bathtub or sink filled to a height h with water. At time $t = 0$, the drain is opened and water flows out. What is the subsequent height of the water?

At $t = 0$, the water level and therefore the pressure is at its highest, so the water drains most rapidly at $t = 0$. As the water drains and the level falls, the pressure and the rate of drainage also fall. This behavior is captured by the following differential equation:

$$\frac{dh}{dt} = -f(h),$$

where $f(h)$ is an as-yet-unknown function of the height.

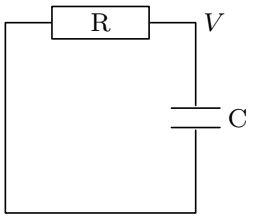
Finding $f(h)$ requires knowing the geometry of the tub and piping and then calculating the flow resistance in the drain and piping. The simplest model for resistance is a so-called linear leak: that $f(h)$ is proportional to h. Then the differential equation simplifies to

$$\frac{dh}{dt} \propto -h.$$

What are the dimensions of the missing constant of proportionality?

The derivative on the left side has dimensions of speed (height per time), so the missing constant has dimensions of inverse time. Call the constant $1/\tau$, where τ is the **time constant** of the system. Then

$$\frac{dh}{dt} = -\frac{h}{\tau}.$$



An almost-identical differential equation describes the voltage V on a capacitor discharging across a resistor:

$$\frac{dV}{dt} = -\frac{1}{RC} V.$$

It is the leaky-tank differential equation with time constant $\tau = RC$.

Problem 1.3 Kirchoff's laws
Use Kirchoff's laws to verify this differential equation.

Approximating the continuous-time differential equation as a discrete-time system enables the system to be simulated by hand and computer. In a discrete-time system, time advances in lumps.

If the lump size, also known as the timestep, is T, then $h[n]$ is the discrete-time approximation of $h(nT)$. Imagine that the system starts with $h[0] = h_0$. What is $h[1]$? In other words, what is the discrete-time approximation for $h(T)$? The leaky-tank equation says that

$$\frac{dh}{dt} = -\frac{h}{\tau}.$$

At $t = 0$ this derivative is $-h_0/\tau$. If dh/dt stays fixed for a whole timestep – a slightly dubious but simple assumption – then the height falls by approximately $h_0 T/\tau$ in one timestep. Therefore

$$h[1] = h_0 - \frac{T}{\tau} h_0 = \left(1 - \frac{T}{\tau}\right) h[0].$$

▶ *Using the same assumptions, what is* $h[2]$ *and, in general,* $h[n]$*?*

The reasoning to compute $h[1]$ from $h[0]$ applies when computing $h[2]$ from $h[1]$. The derivative at $n = 1$ – equivalently, at $t = T$ – is $-h[1]/\tau$. Therefore between $n = 1$ and $n = 2$ – equivalently, between $t = T$ and $t = 2T$ – the height falls by approximately $-h[1]T/\tau$,

$$h[2] = h_1 - \frac{T}{\tau} h_1 = \left(1 - \frac{T}{\tau}\right) h[1].$$

This pattern generalizes to a rule for finding every $h[n]$:

$$h[n] = \left(1 - \frac{T}{\tau}\right) h[n-1].$$

This implicit equation has the closed-form solution

$$h[n] = h_0 \left(1 - \frac{T}{\tau}\right)^n.$$

▶ *How closely does this solution reproduce the behavior of the original, continuous-time system?*

The original, continuous-time differential equation $dh/dt = -h/\tau$ is solved by

$$h(t) = h_0 e^{-t/\tau}.$$

At the discrete times $t = nT$, this solution becomes

$$h(t) = h_0 e^{-nT/\tau} = h_0 \left(e^{-T/\tau}\right)^n.$$

The discrete-time approximation replaces $e^{-T/\tau}$ with $1 - T/\tau$. That difference is the first two terms in the Taylor series for $e^{-T/\tau}$:

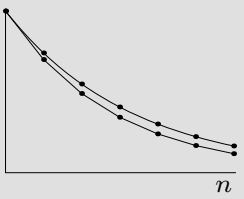
$$e^{-T/\tau} = 1 - \frac{T}{\tau} + \frac{1}{2}\left(\frac{T}{\tau}\right)^2 - \frac{1}{6}\left(\frac{T}{\tau}\right)^3 + \ldots.$$

Therefore the discrete-time approximation is accurate when the higher-order terms in the Taylor series are small – namely, when $T/\tau \ll 1$.

This method of solving differential equations by replacing them with discrete-time analogs is known as the Euler approximation, and it can be used to solve equations that are very difficult or even impossible to solve analytically.

Problem 1.4 Which is the approximate solution?

Here are unlabeled graphs showing the discrete-time samples $h[n]$ and the continuous-time samples $h(nT)$, for $n = 0 \ldots 6$. Which graph shows the discrete-time signal?



Problem 1.5 Large timesteps

Sketch the discrete-time samples $h[n]$ in three cases: (a.) $T = \tau/2$ (b.) $T = \tau$ (c.) $T = 2\tau$ (d.) $T = 3\tau$

Problem 1.6 Tiny timesteps

Show that as $T \to 0$, the discrete-time solution

$$h[n] = h_0 \left(1 - \frac{T}{\tau}\right)^n.$$

approaches the continuous-time solution

$$h(t) = h_0 e^{-nT/\tau}.$$

How small does T have to be, as a function of n, in order that the two solutions approximately match?

1.3 Fall of a fog droplet

The leaky tank (Section 1.2) is a first-order system, and its differential equation and difference equation are first-order equations. However, the physical world is often second order because Newton's second law of motion, $F = ma$, contains a second derivative.

For such systems, how applicable is the Euler approximation? To illustrate the issues that arise in applying the Euler approximation to second-order systems, let's simulate the fall of a fog droplet acted on by gravity ($F = mg$) and air resistance. A fog droplet is small enough that its air resistance is proportional to velocity rather than to the more usual velocity squared. Then the net downward force on the droplet is $mg - bv$, where v is its velocity and b is a constant that measures the strength of the drag. In terms of position x, with the positive direction as downward, Newton's second law becomes

$$m\frac{d^2x}{dt^2} = mg - b\frac{dx}{dt}.$$

Dividing both sides by m gives

$$\frac{d^2x}{dt^2} = g - \frac{b}{m}\frac{dx}{dt}.$$

▶ *What are the dimensions of* b/m*?*

The constant b/m turns the velocity dx/dt into an acceleration, so b/m has dimensions of inverse time. Therefore rewrite it as $1/\tau$, where $\tau \equiv m/b$ is a time constant. Then

$$\frac{d^2x}{dt^2} = g - \frac{1}{\tau}\frac{dx}{dt}.$$

▶ *What is a discrete-time approximation for the second derivative?*

In the leaky-tank equation,

$$\frac{dh}{dt} = -\frac{h}{\tau},$$

the first derivative at $t = nT$ had the Euler approximation $(h[n+1] - h[n])/T$ and $h(t = nT)$ became $h[n]$. The second derivative d^2x/dt^2 is the limit of a difference of two first derivatives. Using the Euler approximation procedure, approximate the first derivatives at $t = nT$ and $t = (n+1)T$:

$$\left.\frac{dx}{dt}\right|_{t=nT} \approx \frac{x[n+1] - x[n]}{T};$$
$$\left.\frac{dx}{dt}\right|_{t=(n+1)T} \approx \frac{x[n+2] - x[n+1]}{T}.$$

Then

$$\left.\frac{d^2x}{dt^2}\right|_{t=nT} \approx \frac{1}{T}\left(\frac{x[n+2] - x[n+1]}{T} - \frac{x[n+1] - x[n]}{T}\right).$$

This approximation simplifies to

$$\left.\frac{d^2x}{dt^2}\right|_{t=nT} \approx \frac{1}{T^2}\left(x[n+2] - 2x[n+1] + x[n]\right).$$

The Euler approximation for the continuous-time equation of motion is then

$$\frac{1}{T^2}\left(x[n+2] - 2x[n+1] + x[n]\right) = g - \frac{1}{\tau}\left(\frac{x[n+1] - x[n]}{T}\right)$$

or

$$x[n+2] - 2x[n+1] + x[n] = gT^2 - \frac{T}{\tau}(x[n+1] - x[n]).$$

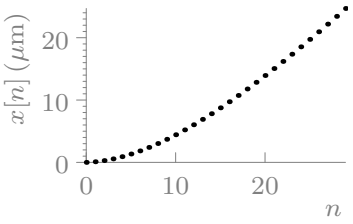
Our old friend from the leaky tank, the ratio T/τ, has reappeared in this problem. To simplify the subsequent equations, define $\alpha \equiv T/\tau$. Then after collecting the like terms, the difference equation for the falling fog droplet is

$$x[n+2] = (2 - \alpha)x[n+1] - (1 - \alpha)x[n] + gT^2.$$

As expected, this difference equation is second order. Like the previous second-order equation, the Fibonacci equation, it needs two initial values. Let's try $x[-1] = x[0] = 0$. Physically, the fog droplet starts from rest at the reference height 0, and at $t = 0$ starts feeling the gravitational force mg.

For a typical fog droplet with radius 10 μm, the physical parameters are:

$$\begin{aligned} m &\sim 4.2 \cdot 10^{-12}\,\mathrm{kg};\\ b &\sim 2.8 \cdot 10^{-9}\,\mathrm{kg\,s^{-1}};\\ \tau &\sim 1.5 \cdot 10^{-3}\,\mathrm{s^{-1}}. \end{aligned}$$



Relative to τ, the timestep T should be small, otherwise the simulation error will large. A timestep such as 0.1 ms is a reasonable compromise between keeping reducing the error and speeding up the simulation. Then the dimensionless ratio α is 0.0675. A simulation using these parameters shows x initially rising faster than linearly, probably quadratically, then rising linearly at a rate of roughly 1.5 μm per timestep or $1.5\,\mathrm{cm\,s^{-1}}$.

This simulation result explains the longevity of fog. Fog is, roughly speaking, a cloud that has sunk to the ground. Imagine that this cloud reaches up to 500 m (a typical cloud thickness). Then, to settle to the ground, the cloud requires

$$t_{\text{fall}} \sim \frac{500\,\mathrm{m}}{1.5\,\mathrm{cm\,s^{-1}}} \sim 9\,\text{hours}.$$

In other words, fog should last overnight – in agreement with experience!

Counting timesteps How many timesteps would the fog-droplet simulation require (with $T = 0.1$ ms) in order for the droplet to fall 500 m in the simulation? How long would your computer, or another easily available computer, require to simulate that many timesteps?

Problem 1.7 Terminal velocity

By simulating the fog equation

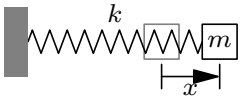
$$x[n+2] = (2-\alpha)x[n+1] - (1-\alpha)x[n] + gT^2.$$

with several values of T and therefore α, guess a relation between g, T, α, and the terminal velocity of the particle.

1.4 Springs

Now let's extend our simulations to the most important second-order system: the spring. Springs are a model for a vast number of systems in the natural and engineered worlds: planetary orbits, chemical bonds, solids, electromagnetic radiation, and even electron–proton bonds. Since color results from electromagnetic radiation meeting electron–proton bonds, grass is green and the sky is blue because of how springs interact with springs.

The simplest spring system is a mass connected to a spring and free to oscillate in just one dimension. Its differential equation is



$$m\frac{d^2x}{dt^2} + kx = 0,$$

where x is the block's displacement from the equilibrium position, m is the block's mass, and k is the spring constant. Dividing by m gives

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0.$$

Defining the angular frequency $\omega \equiv \sqrt{k/m}$ gives the clean equation:

$$\frac{d^2x}{dt^2} + \omega^2 x = 0.$$

Now divide time into uniform steps of duration T, and replace the second derivative d^2x/dt^2 with a discrete-time approximation:

$$\left.\frac{d^2x}{dt^2}\right|_{t=nT} \approx \frac{x[n+2] - 2x[n+1] + x[n]}{T^2},$$

where as usual the sample $x[n]$ corresponds to the continuous-time signal $x(t)$ at $t = nT$. Then

$$\frac{x[n+2] - 2x[n+1] + x[n]}{T^2} + \omega^2 x[n] = 0$$

or after collecting like terms,

$$x[n+2] = 2x[n+1] - \left(1 + (\omega T)^2\right)x[n].$$

Defining $\alpha \equiv \omega T$,

$$x[n+2] = 2x[n+1] - \left(1 + \alpha^2\right)x[n].$$

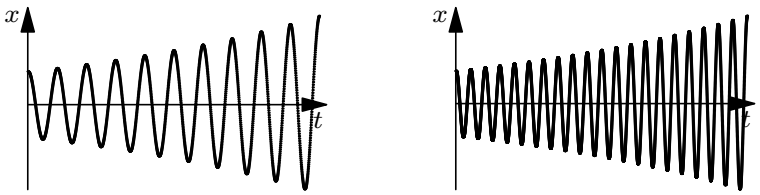
This second-order difference equation needs two initial values. A simple pair is $x[0] = x[1] = x_0$. This choice corresponds to pulling the mass rightwards by x_0, then releasing it at $t = T$. What happens afterward?

To simulate the system numerically, one should choose T to make α small. As a reasonable small α, try 100 samples per oscillation period: $\alpha = 2\pi/100$ or roughly 0.06. Alas, the simulation predicts that the oscillations grow to infinity.

```
name: dummy
file: shm-forward
state: unknown
```

▶ *What went wrong?*

Perhaps α, even at 0.06, is too large. Here are two simulations with smaller values of α:



$\alpha \approx 0.031$ $\alpha \approx 0.016$

These oscillations also explode. The only difference seems to be the rate of growth (Problem 1.8).

Problem 1.8 Tiny values of α
Simulate

$$x[n+2] = 2x[n+1] - (1+\alpha^2)x[n]$$

using very small values for α. What happens?

An alternative explanation is that the discrete-time approximation of the derivative caused the problem. If so, it would be surprising, because the same approximation worked when simulating the fall of a fog droplet. But let's try an alternative definition: Instead of defining

$$\frac{dx}{dt}\Big|_{t=nT} \approx \frac{x[n+1] - x[n]}{T},$$

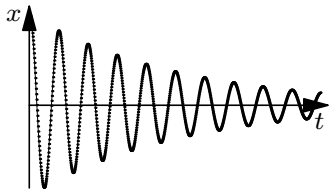
try the simple change to

$$\frac{dx}{dt} \approx \frac{x[n] - x[n-1]}{T}.$$

Using the same procedure for the second derivative,

$$\left.\frac{d^2x}{dt^2}\right|_{t=nT} \approx \frac{x[n] - 2x[n-1] + x[n-2]}{T^2}.$$

The discrete-time spring equation is then



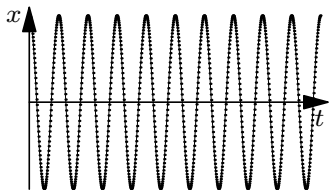
$$(1 + \alpha^2)x[n] = 2x[n-1] - x[n-2],$$

or

$$x[n] = \frac{2x[n-1] - x[n-2]}{1 + \alpha^2}.$$

Using the same initial conditions $x[0] = x[1] = 1$, what is the subsequent time course? The bad news is that these oscillations decay to zero!

However, the good news is that changing the derivative approximation can significantly affect the behavior of the discrete-time system. Let's try a symmetric second derivative:



$$\left.\frac{d^2x}{dt^2}\right|_{t=nT} \approx \frac{x[n+1] - 2x[n] + x[n-1]}{T^2}.$$

Then the difference equation becomes

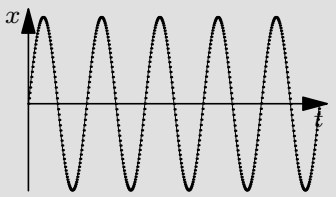
$$x[n+2] = (2 - \alpha^2)x[n+1] - x[n].$$

Now the system oscillates stably, just as a spring without energy loss or input should behave.

Why did the simple change to a symmetric second derivative solve the problem of decaying or growing oscillations? The representation of the alternative discrete-time systems as difference equations does not help answer that question. Its answer requires the two most important ideas in signals and systems: operators (??) and modes (??).

Problem 1.9 Different initial conditions

Here are the subsequent samples using the symmetric second derivative and initial conditions $x[0] = 0$, $x[1] = x_0$. The amplitude is, however, much larger than x_0. Is that behavior physically reasonable? If yes, explain why. If not, explain what should happen.



2
Difference equations and modularity

2.1 Modularity: Making the input like the output 17
2.2 Endowment gift 21
2.3 Rabbits 25

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 Fact-Book* [2], 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 1. 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 1. 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 2. 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

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

Exercise 2. 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 1% 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 $\varpi 10^7$ 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 $\varpi 0.5 \times 10^6$ 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 [9]. 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 3. 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 4. 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

$\delta = 0.01/year$. 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 5. 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 6. 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 3. 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 7. 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 8. 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 4. 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 5. 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 6. 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 [6]:

> 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 9. 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 10. 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 11. 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* [10], 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 7. Write the corresponding Matlab or Octave code, then rewrite the code in one of the languages – Python, Matlab, or Octave – to be efficient.

Exercise 8. 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 12. 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 9. Use a more refined argument to improve the upper bound to $z < \sqrt{3}$.

Exercise 10. Does the number 1.618 look familiar?

Exercise 11. [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 $z \approx 1.618$?

3

Block diagrams and operators: Two new representations

3.1 Disadvantages of difference equations 34
3.2 Block diagrams to the rescue 35
3.3 The power of abstraction 40
3.4 Operations on whole signals 41
3.5 Feedback connections 45
3.6 Summary 49

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.

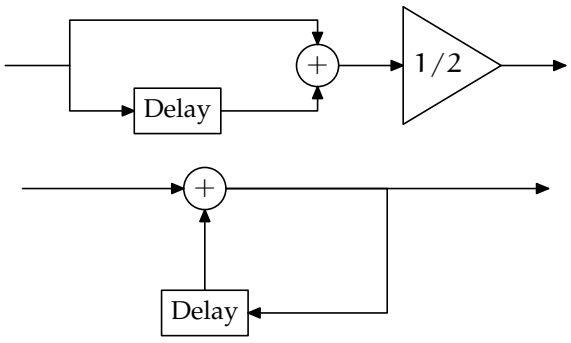
Exercise 12. Although this section pointed out a disadvantage of difference equations, it is also important to appreciate their virtues. Therefore, invent a verbal description (a story) to represent the single equation

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

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:

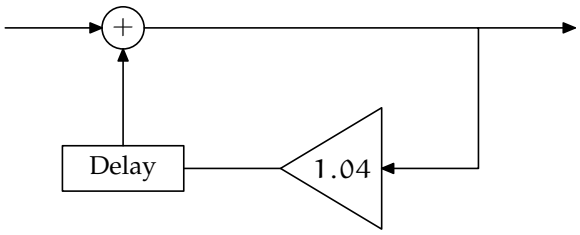


$$y[n] = (x[n] + x[n-1])/2$$
averaging filter

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

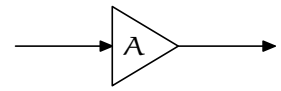
Pause to try 13. Draw the block diagram for the endowment account from Section 2.2.

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:



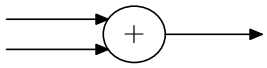
$y[n] = 1.04\,y[n-1] + x[n]$
endowment account from Section 2.2

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 14.* Represent the Fibonacci system of Section 1.1 using a block diagram.

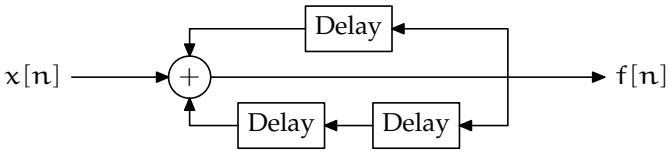
We could translate Fibonacci's description (Section 1.1) directly into a block diagram, but we worked so hard translating the description into a difference equation that we start there. Its difference equation is

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

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 13. 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

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

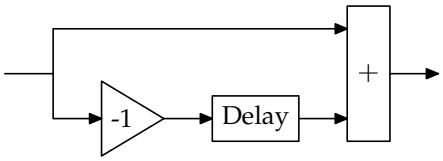
is equivalent to

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

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

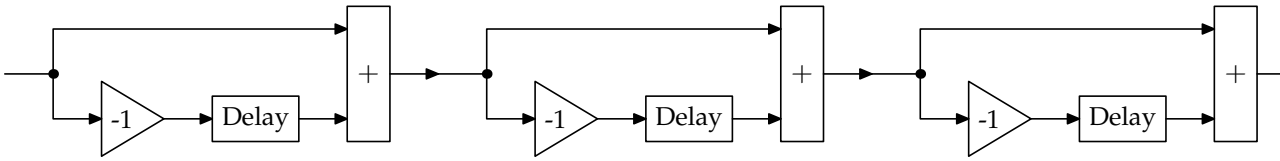
$$\text{output} = \text{this input} - \text{previous input}.$$

The block diagram for that structure is



where the gain of -1 produces the subtraction.

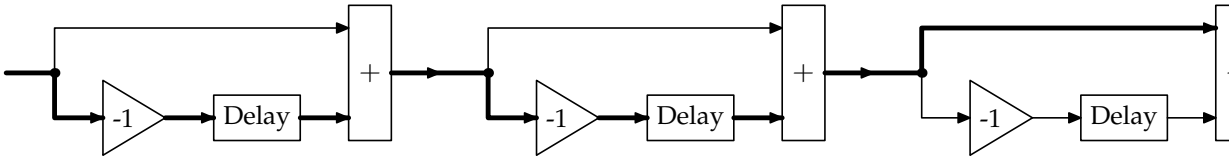
The cascade of three such structures has the block diagram



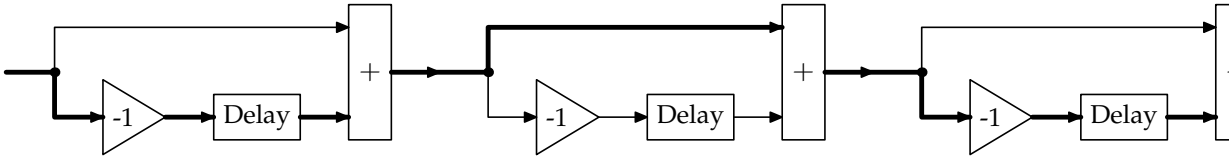
This diagram has advantages compared to the set of difference equations. First, the diagram helps us describe the system compactly. Each stage in the cascade is structurally identical, and the structural identity is apparent by looking at it. Whereas in the difference-equation representation, the common 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:

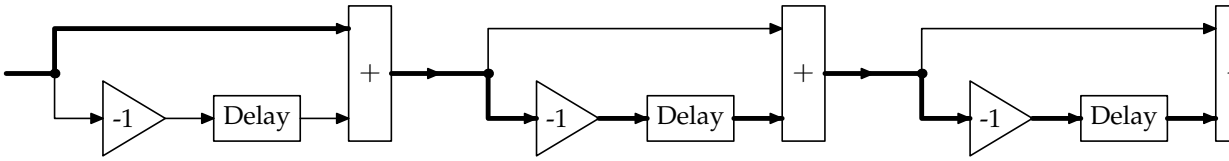
1. low road, low road, high road:



2. low road, high road, low road:



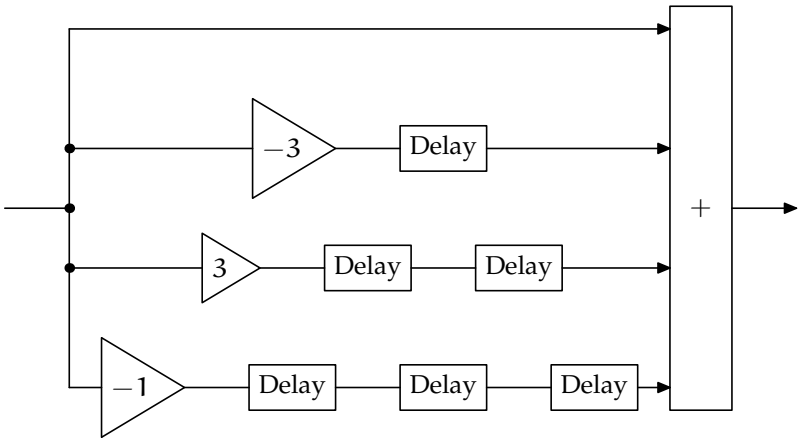
3. high road, low road, low road:



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 14. 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 [7] 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 [11] or the wonderful textbook *Spacetime Physics* [1], 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.

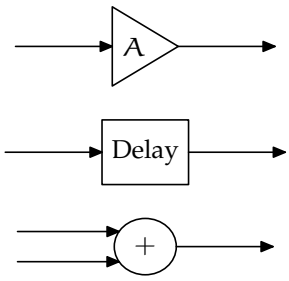
Exercise 15. Find a few abstractions in chemistry, biology, physics, and programming.

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 [15].

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 15. 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 16.* What are the operator versions of the three difference equations?

The system of equations turns into these operator expressions

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

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

Exercise 17. What is the result of applying $(1 - \mathcal{R})^2$ to the signal $1, 4, 9, 16, 25, 36, \ldots$?

3.5 Feedback connections

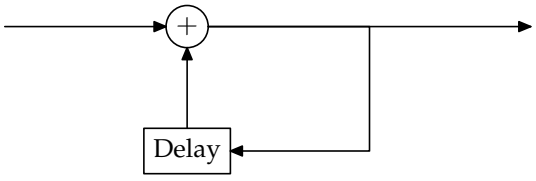
The system with $(1 - \mathcal{R})^3$ as its system functional used only feedforward connections: The output could be computed directly from a fixed number of inputs. However, many systems – such as Fibonacci or bank accounts – contain feedback, where the output depends on previous values of the output.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

3.5.1 Accumulator

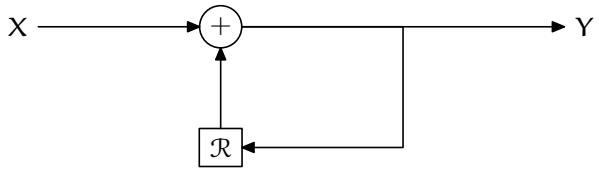
Here is the difference equation for the simplest feedback system, an **accumulator**:

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

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 17. 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 18. 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, $\mathcal{R}$, $\mathcal{R}^2$, ... to the impulse I:

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

Add these signals to get the output signal Y.

Pause to try 18. What is Y?

For $n \geqslant 0$, the $y[n]$ sample gets a 1 from the $\mathcal{R}^nI$ 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 19. 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:

$$\text{output} = \text{delayed output} + \text{twice-delayed output} + \text{input}.$$

Pause to try 19. 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 20. 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.

4
Modes

4.1 Growth of the Fibonacci series 52
4.2 Taking out the big part from Fibonacci 55
4.3 Operator interpretation 57
4.4 General method: Partial fractions 59

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 [12]. 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 [16], 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 [4, 14, 8].

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 1.1.2 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 21. 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 20. 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 .
By using the Inverse Symbolic Calculator, you increase your repertoire of closed form and thereby enhance your intuition.

Pause to try 21. 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 22. 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 22. 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 23. 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:

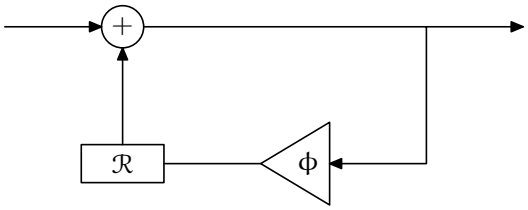
$$y[n] = 7y[n-1] - 12y[n-2] + x[n].$$

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

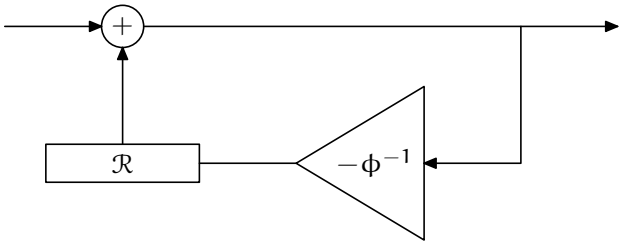
make, and are the building blocks of all systems, so we would like to find the operator or block-diagram representations for a mode.

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



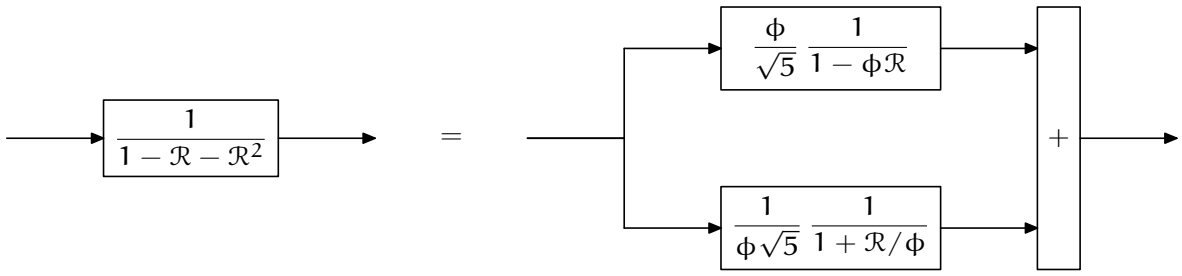
with the system functional $(1 + \mathcal{R}/\phi)^{-1}$.

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:



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

Exercise 24. 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 25. 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?

5
Repeated roots

5.1 Leaky-tank background 64
5.2 Numerical computation 65
5.3 Analyzing the output signal 67
5.4 Deforming the system: The continuity argument 68
5.5 Higher-order cascades 70

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 (??) or the leaky tank (Section 1.2). 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 26. 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

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

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

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

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

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

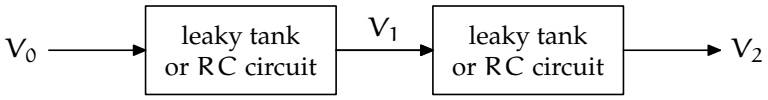
Pause to try 23. What system functional corresponds to this difference equation?

The corresponding system functional is

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

Exercise 27. 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 24. 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 n^{th} 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

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

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 28. 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 29. 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 30. Sketch $n+1$, 0.95^n, and their product.

Pause to try 25. 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 31. Guess the quadratic prefactor of the impulse response of the triple cascade. How can you confirm your guess?

Exercise 32. Compute and sketch the output signal of the triple cascade.

Exercise 33. Where is the maximum of the impulse response? How does it compare to the maximum for the double cascade?

6
The perfect (sine) wave

6.1 Forward Euler 72
6.2 Backward Euler 76
6.3 Leapfrog 79
6.4 Summary 82

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:

$$\dot{y}_1 = y_2,$$
$$\dot{y}_2 = -y_1.$$

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

$$\dot{x} = v,$$
$$a = -x.$$

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

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

Then the continuous-time system 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.

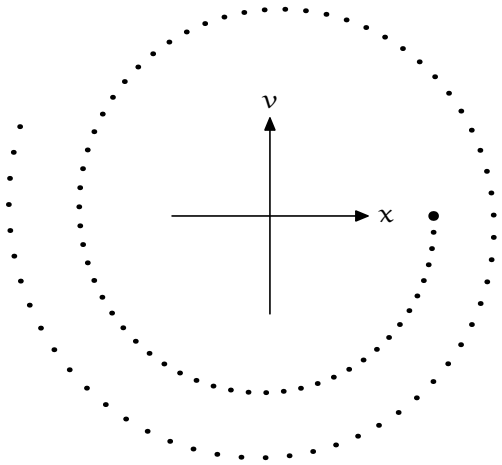
6.1.1 Simulation

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 26. 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

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

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

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

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 27. 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

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

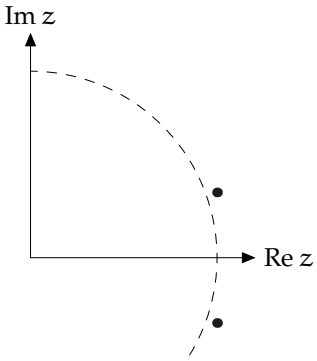
The sum of the roots is 2 while the product is greater than 1.

Pause to try 28. 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 34. To place the poles on the unit circle, why not simulate with $T = 0$?

Exercise 35. 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:

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

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

Then the system of continuous-time equations becomes

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

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

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

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

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

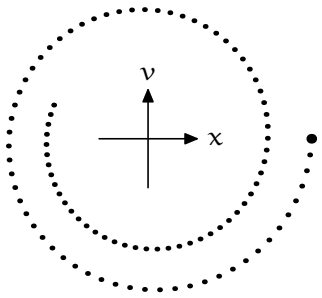
These equations are similar to the forward-Euler equations except for the factors of $1 + T^2$. Those factors shrink $x[n]$ and $v[n]$, so they might control the runaway oscillations.

Pause to try 29. 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 30.* 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

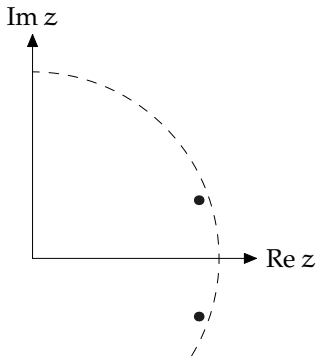
$$(1 + T^2)x[n] - 2x[n-1] + x[n-2] = f[n]$$

and this system functional

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

> *Pause to try 31.* 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 32.* 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

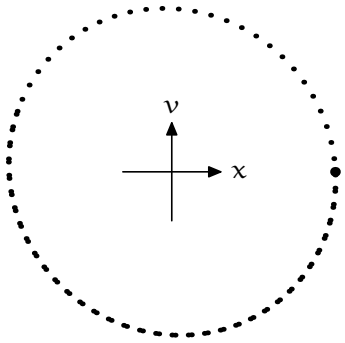
$$x[n+1] = x[n] + Tv[n],$$
$$v[n+1] = v[n] - Tx[n+1].$$

Pause to try 33. 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:

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

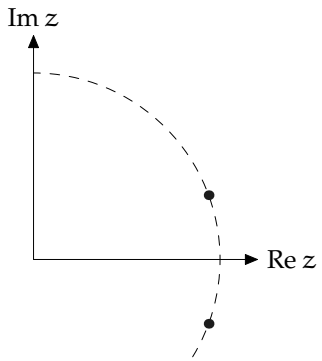
After rearranging and including a forcing function, the result is

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

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 36. 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 [13].

7
Control

7.1 Motor model with feedforward control 83
7.2 Simple feedback control 85
7.3 Sensor delays 87
7.4 Inertia 90

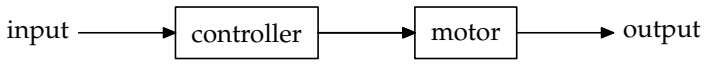
The goals of this chapter are to study:

- 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



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.

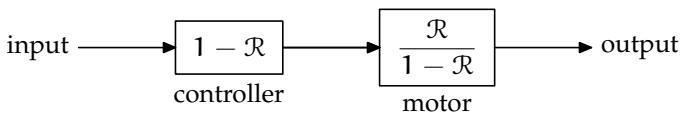
To design the controller, we need to model the motor. The motor turns a voltage into the arm's angular velocity ω. The continuous-time system that turns ω into angle is $\theta \propto \int \omega \, dt$. Its forward-Euler approximation is the difference equation

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

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

Exercise 37. 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.

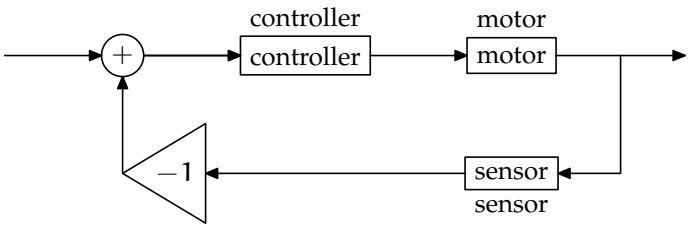
A problem in the same category is an error in the constant of proportionality. Suppose that the motor model underestimates the conversion between voltage and angular velocity, say by a factor of 1.5. Then the system functional of the controller–motor system is $1.5\mathcal{R}$ rather than $\mathcal{R}$. 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

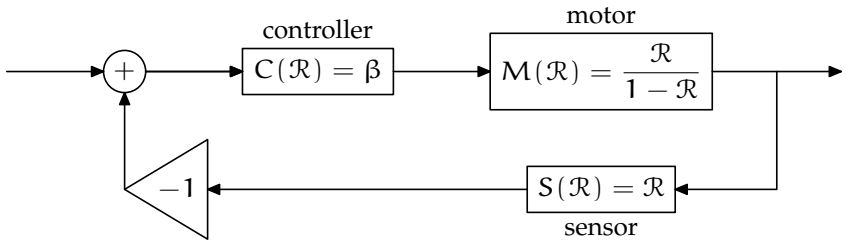
In feedback control, the controller uses the output signal to decide what to tell the motor. Knowing the input and output signals, an infinitely intelligent controller could deduce how the motor works. Such a controller would realize that the arm's angle starts with an offset or that the motor's conversion 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



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 $p_1 p_2 = \beta$. 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 $\beta = p_1 p_2$, 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 38. 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 $S(R) = \mathcal{R}^2$.

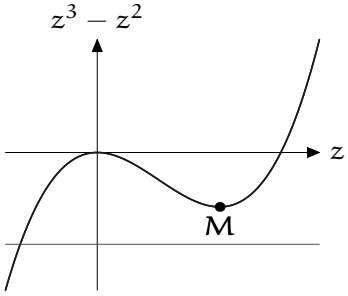
Pause to try 34. With this sensor, what is the functional for the feedback system?

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 35. 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 [18] 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 $p_1 p_2 p_3$ 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 36. 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 $p_1 p_2 p_3 = -\beta$. 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 39. 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 37. 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 38. 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 40. Sketch how the poles move as β changes from 0 to ∞.

Exercise 41. 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 $1/2$.

8

Proportional and derivative control

8.1 Why derivative control 95
8.2 Mixing the two methods of control 96
8.3 Optimizing the combination 98
8.4 Handling inertia 99
8.5 Summary 103

The goals of this chapter are:

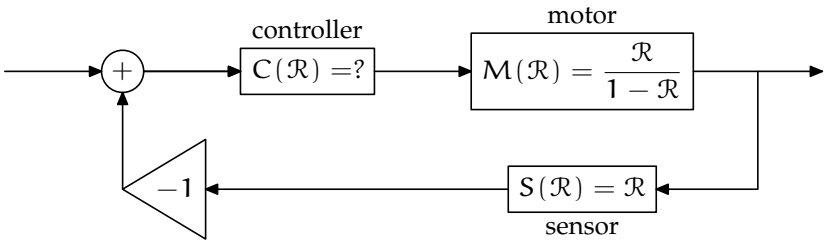
- to introduce derivative control; and
- to study the combination of proportional and derivative control for taming systems with integration or inertia.

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:

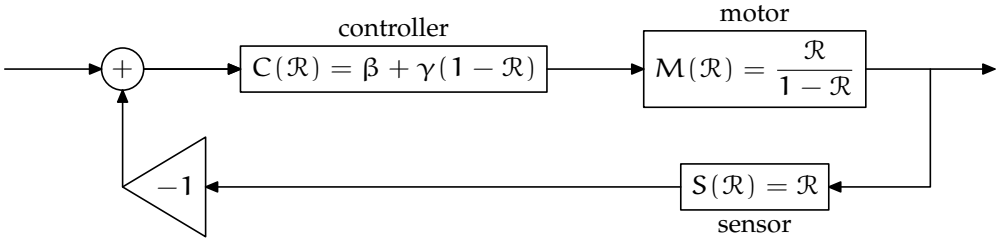


Since the motor has the functional $\mathcal{R}/(1-\mathcal{R})$, let's put a discrete-time derivative $1-\mathcal{R}$ into the controller to remove the $1-\mathcal{R}$ in the motor's denominator. With this **derivative control**, the forward-path cascade of the controller and motor contains only powers 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

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



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

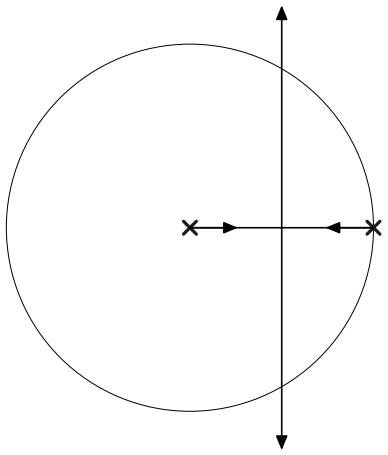
$$F(\mathcal{R}) = \frac{\text{whatever}}{1 - \mathcal{R} + (\beta + \gamma(1-\mathcal{R}))\mathcal{R}^2}.$$

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

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

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 42. 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 43. 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:

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

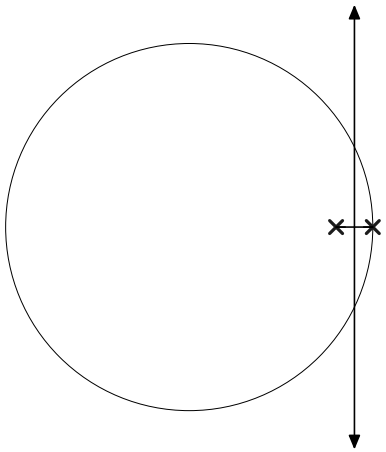
where m is a constant between 0 (no inertia) and 1 (maximum inertia). This term changes the motor's system functional to

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

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.



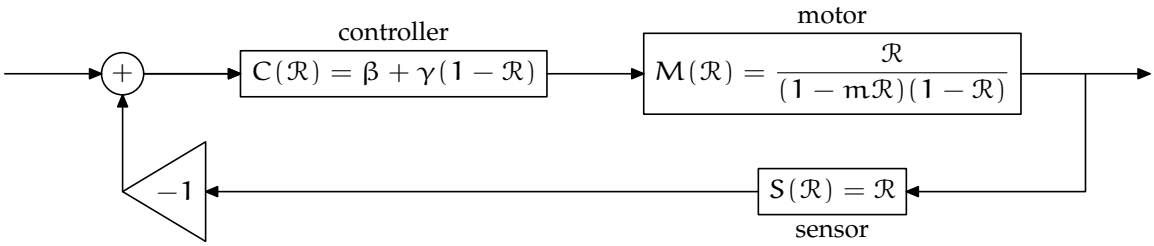
Pause to try 39. How many time steps before 0.9^n has decayed roughly by a factor of e^3 (commonly used as a measure of 'has fallen very close to zero')?

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



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.

Bibliography

[1] *Spacetime Physics*. W. H. Freeman and Co., 1992.

[2] Central Intelligence Agency. *The World Factbook*. Central Intelligence Agency, 2007. https://www.cia.gov/library/publications/the-world-factbook/.

[3] B. S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13(6):4–16, 1984.

[4] Jonathan Borwein and David Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st century*. A K Peters, 2003.

[5] Richard A. Dunlap. *The Golden Ratio and Fibonacci Numbers*. World Scientific, 1997.

[6] Leah Edelstein-Keshet. *Mathematical models in biology*. SIAM, Philadelphia, 2005.

[7] Albert Einstein. Zur elektrodynamik bewegter kÃűrper [On the electrodynamics of moving bodies]. *Annalen der Physik*, 17:891–921, 1905.

[8] David Epstein and Sylvio Levy. Experimentation and proof in mathematics. *Notices of the American Mathematical Society*, pages 670–674, June/July 1995.

[9] Richard Feynman and Ralph Leighton (contributor). *Surely You're Joking, Mr. Feynman! Adventures of a Curious Character*. W. W. Norton, 1985.

[10] Fibonacci. *Liber Abaci.*, 1202.

[11] Hermann Minkowski H. A. Lorentz Albert Einstein and Hermann Weyl. *The Principle of Relativity: A Collection of Original Memoirs*. Dover, 1952.

[12] 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.

[13] Jan Brett (illustrator). *Goldilocks and the Three Bears*. Dodd, 1987.

[14] David Bailey Jonathan Borwein and Roland Girgensohn. *Experimentation in Mathematics: Computational Paths to Discovery*. A K Peters, 2004.

[15] 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.

[16] P. Ribenboim. *The New Book of Prime Number Records*. Springer–Verlag, New York, 1996.

[17] William McC. Siebert. *Circuits, Signals, and Systems*. MIT Press, Cambridge, MA, 1986.

[18] Tjalling J. Ypma. Historical development of the Newton–Raphson method. *SIAM Review*, 37(4):531-551,, 1995.

Index

Note: An italic page number refers to a problem on that page.

~ 24
RC circuit 64

abstraction 40
  whole-signal 33, 41
acceleration 71
accumulator 45
aggressive 82
analogy 45
analysis
  sample by sample 41
angular velocity 84
approximation
  discrete-space 70
Aristotle x
artificial intelligence 85
associative array
  `awk` 53

backward Euler 76
Binet formula 57
binomial theorem 47, 69
black box 51
block diagram
  elements in operator notation 43
  operator 45
block diagrams 33
boundary conditions 17
brick wall 85
buffer 64

calculus
  finite differences, of 18
change
  loose 25
chunks 41
clearing fractions 86
closed form 4, 29, 52
code
  Python 29
compounding
  annual 21
conjectures 67
continuity argument 63, 67, 68
controller 83
control variable 83
convolution 67
coupled oscillator 71
courage 24
cross multiplying 60
curly braces 42

danger 85
data
  playing with,play 67
deforming systems 68
delay element 42
dendrite 70
derivative
  continuous time 72
derivative control 96
derivatives 72
desert island 53

design 19
dictionary
  Python 53
difference equation 17
differentiator
  continuous time 38
  discrete time 38, 46
dimensions 22
discretization 65
distinctions
  finite or infinite 21
division
  incorrect floating-point 52
donor 21
double root 64
drawdown 21

Einstein, Albert 40
elegance 19
endowment 21
engineering design 83
equation
  first-order difference 23
  second-order difference 4, 28
equation hygiene 64
equivalence
  system 59
error signal 84, 85
experiment 30
experimental mathematics 52
explicit recipe 72
exponent notation 43
extreme-case
  large n 69
extreme case
  small n 69
extreme cases 24
  gain 86

feedback 83
feedback control 85
feedforward 45, 83
Feynman, Richard 21
Fibonacci function
  memoized 53
Fibonacci sequence
  decomposition 51
forcing function 74
forward-Euler approximation 84
forward Euler 64, 72
function
  decreasing 69
  increasing 69
fund 21

gain
  increasing 86
golden ratio 54
grammar 40
  block diagrams 44
graphing calculator 53
growth
  exponential 5, 24, 29
  logarithmic 5, 29
  polynomial 5, 29
  rate 18
guess
  solution to a difference equation 24
guess and check 60
guessing 70

hash
  Perl 53

implicit Euler 76
implicit recipe 76
impulse 28
input signal
  arbitrary 23
insight 33
instability 86
intuition 21
Inverse Symbolic Calculator 54

language 40
leaky tank 64
leapfrog 79
left-shift operator 42
letter
  capital 23
  lowercase 23

like terms 64
linear combination 72
linear equations 60
loop gain 96

mathematical definition 71
mathematical translation
  incomplete 25
matrix inversion 76
maxim 21
meaningless objects 52
mind
  amazing feats 40
Minkowski, Hermann 40
MIT 21
mode 51
  amplitude 55
  shape 55
model
  population growth 19
modes 63
modular formulation 19
modularity 17, 23
multiple representations 33
mythical controller 85

negative contribution 24
negative delay 84
notation
  entire signal 23
  one sample 23
number theory 52

operator notation 42
operator representation 42
operators 33, 57
oscillations 30
output signal 22

parameter sensitivity 85
partial fractions 59, 68
party
  graduation 25
passive 82
pattern 67
peeling away 55
philosophy 18
physical meaning 71
pole
  farthest 87
poles 74
  complex 71
population
  growth 17
  United States 18
probe 52
probes
  computational 51
product
  increasing with decreasing function
    69
programming
  object-oriented 20
proportional control 86
Python 53

quadratic formula 75

rabbits 25
  system 26
ratio
  dimensionless 64
recurrence relation 18
relativity
  special 40
repeated root 63
representation 22
  mathematical 17, 22
  operator 34
residual signal 56
right-shift operator 42
robot 85

sensor
  real 85
shift
  left 76
signals and systems 20, 26
simulation 20, 65, 66
simulation data 67
sine wave 71
space 40

spacetime 40
spiral 73
spreadsheet 53
spring
  ideal 71
stability 86
step function 46, 47
successive ratios 52
symmetry 76
synthetic division 47
system
  coupled 74
  first-order 23
  second-order 28
system characterization 52
system functional 44
  with feedback 45

taking out the big part 67
Taylor series 47
tea 84
techniques
  take out the big part 55
time 40
time constant 8, 64
time machine 86
time travel 84
translate
  derivatives 72
trapezoidal 79
tutorial teaching x
twin-prime conjecture 52

unit circle 71, 87
unit sample 28
unknowns
  two 24

variables
  eliminating 27
voltage 84
volume elements 20

warmup 22
Wheeler, John 21

6.003 Signals and Systems
Spring 2010