Mathematics, Reconnected: Volume 2, Continuous Mathematics

Chris Collins
Version 0.1, 2026-10-04

Table of Contents

How to Use This Book

The Knowledge Map

Part V: Calculus

- Chapter 16. Limits and Continuity
- Chapter 17. The Derivative
- Chapter 18. Using Derivatives: Optimisation, Approximation, and Newton's Method
- Chapter 19. The Integral
- Chapter 20. Multivariable Calculus and the Gradient

Part VII: Linear Algebra

- Chapter 26. Vectors
- Chapter 27. Matrices as Transformations
- Chapter 28. Eigenvectors, Bases, and Applications

Part IX: Numerical Mathematics

- Chapter 32. Numerical Mathematics: How Computers Actually Compute
- Appendix A: Cheat Sheet
- Appendix B: Glossary
- Appendix C: Notation and Symbols
- Appendix D: The US Mathematics Curriculum
- Appendix E: Interactive Lab Inventory

How to Use This Book

This book is meant to take you from rusty foundational mathematics to the mathematics a strong computer scientist reads and uses every day. Calculus is one milestone on that journey, not the finish line.

Who this book is for

You probably met most of this material once. Maybe it was UK sixth-form "Maths with Mechanics" decades ago, maybe an American Algebra II class. Since then, some of it has stuck and much of it has faded. When you see

$$ax^2 + bx + c = 0$$

it looks familiar, but words like *discriminant*, *vertex* or *completing the square* may not come back straight away.

So the book assumes **prior exposure with substantial gaps and rusty recall**. It won't treat you as someone who has never seen an equation. It also won't wave a section through with "you already know this". Each topic is complete enough that a real gap won't leave you stuck, and the explanations are built so you can move quickly through anything that comes back easily.

There is a second reader in mind: a teenager in the American school system. Every major topic carries a short **US curriculum** note saying roughly where an American student would meet it. If someone tells you "he's doing Algebra II", you'll know what mathematics that covers, and the explanations here can be shared with them directly.

What this book is, and is not

This book is the **canonical knowledge path**, the core of a larger project that can grow to include experiments, Python and Mathematica notebooks, interactive visualisations, and eventually a companion application with mastery tracking.

The book deliberately avoids becoming:

- **a school textbook**, padded out to fill semesters;
- **a formula encyclopedia**, where formulas are known but not understood;
- **a pure mathematics text**, where rigour comes before intuition and application;
- **a programming book in disguise**, since code reinforces the mathematics but never replaces it;
- **a calculus cram course**, which reaches derivatives fast and leaves holes underneath;
- **an endless prerequisite trap**, since you don't need to relearn everything before reaching interesting material;
- **an exercise treadmill**, because mastery matters more than volume.

How each topic is taught

Every substantial topic follows the same arc:

> *Motivation → intuition → formalisation → mechanics.*

The topic starts with a problem worth solving. Next comes what the idea **means** in plain language, then how to state it precisely, and only after that how to calculate with it efficiently. Formulas are **derived when derivation illuminates**, and theorems are **proved when proof teaches**. Procedures are shown **step by step with nothing hidden**, then summarised as algorithms, because much of mathematics really is algorithmic and you already think in algorithms.

The book keeps returning to four representations of the same idea:

words ⟷ symbols ⟷ pictures ⟷ computation

Being able to move between them is much closer to understanding than memorising any one of them.

It also keeps asking two questions: *Where would I use this?* and *When should I use this, and when shouldn't I?* Knowing when a technique fails is part of knowing the technique.

The boxes

You will see the same kinds of box throughout the book. Here is what each one is for.



Refresher boxes

Short definitions of vocabulary and notation, written to be scanned quickly. Together they form the glossary and the cheat sheet in the appendices: the "I remember learning this, remind me what it means" reference.

Example 1. Procedure boxes

Input: a technique you have just understood.

Steps: the technique written as an explicit algorithm.

Output: something you could translate into code, and often the book does.



Watch-out boxes

Failure modes, common mistakes and limitations. These are not footnotes; they are part of understanding the idea.



Interesting boxes

Where a word came from, who solved the problem first, or an unexpected modern use. Kept short.



Interactive Lab boxes

Places where a hands-on experiment would help, such as dragging a point, changing a matrix or picking a starting value. They are collected in Appendix E as the design inventory for a future companion app.

Computer-science connection sidebars

Where the mathematics shows up in computing: algorithms, graphics, databases, machine learning, information retrieval, numerical software.

US curriculum sidebars

Roughly where an American student would meet the topic. Treat these as a cross-reference, not a rule; it varies by state, district and track.

Test out where possible; slow down where necessary

There is no big entrance exam. Instead, each chapter ends with a **mastery check**: a few problems and a **decision rule**. If they are easy, move on. If they are hard, that tells you exactly which foundation needs more time, and the decision rule points you back to the relevant section.

Each chapter also opens with a **Where this chapter sits** sidebar that lists its prerequisites. If a mastery check exposes a gap, follow those links backwards.

Understand, calculate, implement, experiment

Computation in this book follows a fixed order:

1. understand the idea;
2. calculate a few cases by hand;
3. implement it (usually in a few lines of Python);
4. experiment: break it, vary it, plot it.

Asking a computer algebra system for an answer is not understanding. Writing ten lines of Python *after* doing three iterations by hand is a good way to make the understanding stick. All code uses only Python's standard library and NumPy.

The companion code

The `companion/` folder that accompanies this book has one Python folder per chapter (`ch01_numbers/` to `ch33_horizons/`), and each one follows that same order:

- `walkthrough.py` reproduces the chapter's worked examples line by line, with the reason for each step, then implements the procedure and experiments with it, including the ways it fails;
- `exercises.py` holds functions for you to implement, and `test_exercises.py` tells you when you've got them right; `solutions.py` has reference answers with commentary;
- `lab_*.py` builds the chapter's Interactive Labs as interactive plots you can drag and adjust;
- `python3 -m mathbook.mastery chNN` runs the chapter's mastery check as a quiz and tells you which sections to revisit.

A good rhythm is to read a section, run the matching part of the walkthrough, and do the exercises before moving on. See `companion/README.md` for setup.

The three volumes

The book comes in three volumes that follow the knowledge map:

Volume	Contents	Chapters
1: Foundations to Precalculus	Numbers, algebra, geometry, functions, trigonometry, sequences, complex numbers. Everything the other two volumes build on.	1–15 (Parts I–IV)
2: Continuous Mathematics	Calculus, linear algebra, numerical mathematics.	16–20, 26–28, 32 (Parts V, VII, IX)
3: Discrete and Probabilistic Mathematics	Logic, proof, number theory, combinatorics, graphs, probability, statistics, and the Horizons capstone.	21–25, 29–31, 33 (Parts VI, VIII, X)

Volumes 2 and 3 both build on Volume 1 but not on each other, so you can read them in either order or side by side. Chapter numbers are the same in every edition, which is why Volume 2 jumps from Chapter 20 to Chapter 26: a reference such as "Chapter 17" always means the same chapter, and so does the companion folder `ch17_derivatives/`. A reference to a chapter in another volume names the volume, as in "Volume 2, Chapter 17". In the PDF it links to that chapter in the online edition.

You are reading Volume 2. Each volume has the full cheat sheet, notation guide and US-curriculum map, plus a glossary and Interactive Lab inventory covering its own chapters.

How to read the rest of the book

The chapters are numbered, but the book is not a straight line. The Knowledge Map shows how the topics depend on one another. Discrete mathematics can run alongside algebra. Probability can begin early. Linear algebra does not need calculus. Choose your route, use the mastery checks, and go back when you need to.

Building the book

The source is AsciiDoc. From the repository root:

```shell
make            # regenerate all diagrams, then build build/book.html
make html       # just the HTML
make check      # fail on any AsciiDoc warning
```

Mathematics is typeset with LaTeX notation (`:stem: latexmath`) and rendered by MathJax in the HTML output. Diagrams are SVGs generated from the scripts in `diagrams/`, so they can be regenerated and changed.

The Knowledge Map

Mathematics isn't a ladder. It's a web of ideas, each depending on a few others. This chapter is that web: the route-planner for the rest of the book, and the skeleton of the future skill tree.

Why a map rather than a list

School mathematics in the US is usually taught as a sequence of year-long courses:

Pre-Algebra → Algebra I → Geometry → Algebra II → Precalculus → Calculus

That sequence is a scheduling decision, not a law of nature. The real constraint is **dependency**: you can't understand the derivative without limits, and you can't understand limits without functions. But you **can** learn graph theory long before calculus, and elementary linear algebra needs no calculus at all.

The map below shows each chapter and the chapters it builds on directly. An arrow from A to B means "B uses ideas from A". The dashed arrow marks a dependency that only matters for the optional parts of a chapter.

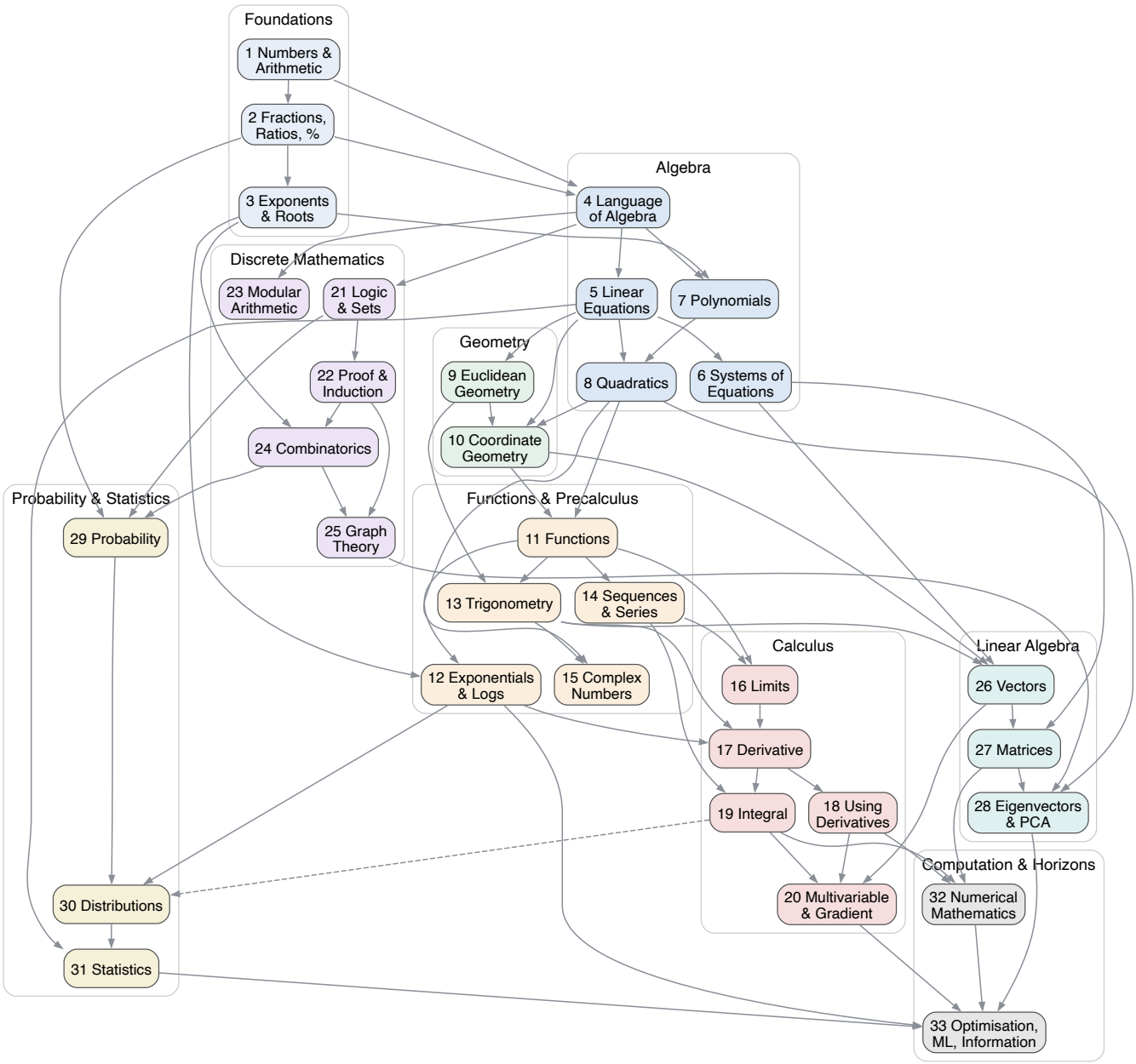


Figure 1. The knowledge map. Arrows point from prerequisite to dependent chapter.

The three volumes cut the map along its natural seams. Volume 1 is the foundation every other chapter rests on (Parts I–IV). Volume 2 holds the continuous branches: calculus, linear algebra and numerical mathematics. Volume 3 holds the discrete and probabilistic branches and ends with the Horizons capstone, which draws on both.

Computer-science connection: this map is a DAG

The map is a *directed acyclic graph*. "Find an order to study the chapters so every prerequisite comes first" is exactly **topological sorting**, the same problem a build system solves when it decides which files to compile first, or a package manager solves when it installs dependencies. You'll implement it in Volume 3, Chapter 25 . The numbering of this book is one valid topological order, but there are many others.

The spine and the branches

The **spine** is the classical route to calculus:

Arithmetic → Algebra → Geometry → Functions → Trigonometry → Precalculus → Calculus

For a computer scientist, several **branches** deserve equal standing:

Branch	Why it matters in computing
Calculus (Chapter 16–Chapter 20)	Continuous change, optimisation, physics simulation, the training of machine-learning models.
Linear algebra (Chapter 26–Chapter 28)	Graphics, transformations, embeddings, search, machine learning.
Probability (Volume 3, Chapter 29 , Volume 3, Chapter 30)	Uncertainty, randomised algorithms, hashing, machine learning.
Statistics (Volume 3, Chapter 31)	Inference, experimentation, A/B testing, metrics.
Logic and sets (Volume 3, Chapter 21)	Programming languages, databases, verification.
Proof (Volume 3, Chapter 22)	Algorithm correctness, loop invariants, reasoning about code.
Number theory (Volume 3, Chapter 23)	Hashing, checksums, cryptography.
Combinatorics (Volume 3, Chapter 24)	Algorithm analysis, probability, complexity.
Graph theory (Volume 3, Chapter 25)	Networks, search, compilers, dependency resolution, distributed systems.
Numerical mathematics (Chapter 32)	How computers actually approximate mathematical answers.

Volume 3, Chapter 33 brings the branches together in optimisation, machine learning and information theory.

Suggested routes

These are starting points, not prescriptions. Use the mastery checks to skip what you already know.

The calculus route

Volume 1, Chapter 1 → Volume 1, Chapter 2 → Volume 1, Chapter 3 → Volume 1, Chapter 4 → Volume 1, Chapter 5 → Volume 1, Chapter 7 → Volume 1, Chapter 8 → Volume 1, Chapter 10 → Volume 1, Chapter 11 → Volume 1, Chapter 12 → Volume 1, Chapter 13 → Volume 1, Chapter 14 → Chapter 16 → Chapter 17 → Chapter 18 → Chapter 19.

The computer-science core

Foundations and Algebra, then Volume 3, Chapter 21 → Volume 3, Chapter 22 → Volume 3, Chapter 23 → Volume 3, Chapter 24 → Volume 3, Chapter 25 → Volume 3, Chapter 29 . These can run alongside the calculus route.

The machine-learning route

The calculus route through Chapter 18, plus Chapter 26 → Chapter 27 → Chapter 28, then Volume 3, Chapter 29 → Volume 3, Chapter 30 → Volume 3, Chapter 31 , then Chapter 20 → Volume 3, Chapter 33 .

The "helping with homework" route

Follow the US-curriculum sidebars. Appendix D maps each school course to the chapters that cover it.

From map to skill tree

Each chapter breaks down into skills. In a future companion app, the map becomes a skill tree whose progress reflects **demonstrated understanding**, not the number of problems ground through:

```
Algebra                          Geometry
  ● Linear equations               ● Coordinates
  ● Quadratics                     ○ Vectors
  ● Functions                      ○ Transformations
  ○ Exponents
  ○ Logarithms
```

Because the tree records dependencies, it can tell you more than "wrong answer". For example: *You understand equations but seem rusty on exponent laws. Refresh those two ideas before logarithms.* That's the purpose of the "Where this chapter sits" sidebar at the top of every chapter and the decision rule at the bottom.

US curriculum

A rough correspondence, detailed in Appendix D:

- **Elementary school**: Volume 1, Chapter 1 , Volume 1, Chapter 2 , early Volume 1, Chapter 9
- **Middle school / Pre-Algebra**: Volume 1, Chapter 2 , Volume 1, Chapter 3 , Volume 1, Chapter 4
- **Algebra I**: Volume 1, Chapter 5 , Volume 1, Chapter 6 , early Volume 1, Chapter 7 , Volume 1, Chapter 8 , Volume 1, Chapter 11
- **Geometry**: Volume 1, Chapter 9 , Volume 1, Chapter 10
- **Algebra II**: Volume 1, Chapter 7 , Volume 1, Chapter 12 , Volume 1, Chapter 15 , Volume 1, Chapter 14
- **Precalculus**: Volume 1, Chapter 11 , Volume 1, Chapter 13 , Volume 1, Chapter 14 , introduction to Chapter 16
- **AP Calculus**: Chapter 16 to Chapter 19
- **AP Statistics**: Volume 3, Chapter 29 , Volume 3, Chapter 30 , Volume 3, Chapter 31
- **University (CS / math)**: discrete mathematics, linear algebra, multivariable calculus, numerical methods

Part V: Calculus

Chapter 16. Limits and Continuity

Chapter map: calculus rests on one idea — the limit — which makes precise what it means for a quantity to "approach" a value. This chapter starts from a concrete puzzle (what does "travelling at 63 mph at this instant" mean?), builds the informal idea of a limit with tables and pictures, then develops the tools for computing limits, the notion of continuity, the intermediate value theorem with its algorithmic twin the bisection method, and limits at infinity. An optional section presents the rigorous ε–δ definition as a game. Throughout, we watch what happens when a computer tries to take limits with floating-point numbers.

Why this chapter matters: a speed camera, a fitness tracker's live pace, the temperature your coffee settles to, the rate of return on an investment, and the claim that one sorting algorithm beats another "for large inputs" are all limits. So is every derivative and integral in the rest of the book. And because computers cannot take limits exactly, the same idea explains a whole family of numerical bugs.

Where this chapter sits

You will need: functions, domains, and graphs (Volume 1, Chapter 11); factoring and rational expressions (Volume 1, Chapter 7); exponentials and logarithms (Volume 1, Chapter 12); radians and the sine function (Volume 1, Chapter 13); sequences and the informal idea of convergence (Volume 1, Chapter 14); and absolute value as distance (Volume 1, Chapter 5).

This chapter unlocks: the derivative (Chapter 17) and everything built on it (Chapter 18), the integral (Chapter 19), multivariable calculus (Chapter 20), the analysis of numerical algorithms (Chapter 32), and the precise meaning of big-O notation.

16.1. The problem: speed at an instant

16.1.1. What does "63 mph right now" mean?

Average speed is easy. If a car covers 120 miles in 2 hours, its average speed is

$$\text{average speed} = \frac{\Delta d}{\Delta t} = \frac{120 \text{ miles}}{2 \text{ hours}} = 60 \text{ mph},$$

where Δ (capital Greek delta) means "change in". But a speedometer claims something stronger: *at precisely this instant*, the car is travelling at 63 mph. What can that mean? At a single instant the car travels zero distance in zero time, and $\frac{0}{0}$ is meaningless. Yet the speedometer reading is clearly not nonsense — a police radar gun and the car agree on it, and physics depends on it.

Here is a concrete version. A car accelerates steadily so that, t hours after setting off, it has travelled

$$d(t) = 21t^2 \text{ miles}.$$

How fast is it going at $t = 1.5$ hours?

16.1.2. Shrink the interval

We cannot compute speed over an interval of zero length, but we can compute average speed over *short* intervals from $t = 1.5$ to $t = 1.5 + h$, and watch what happens as h shrinks: *Try it:* Watch the secants shrink.

PYTHON

```
def d(t):
    """Distance travelled (miles) after t hours."""
    return 21 * t**2

t0 = 1.5
for h in [1, 0.5, 0.1, 0.01, 0.001, 0.0001]:
    avg = (d(t0 + h) - d(t0)) / h
    print(f"h = {h:<7} average speed = {avg:.6f} mph")
```

```
h = 1       average speed = 84.000000 mph
h = 0.5     average speed = 73.500000 mph
h = 0.1     average speed = 65.100000 mph
h = 0.01    average speed = 63.210000 mph
h = 0.001   average speed = 63.021000 mph
h = 0.0001  average speed = 63.002100 mph
```

The averages are homing in on 63. No single interval gives exactly 63 — every one gives a bit more, because the car is speeding up during the interval — but by making h small enough we can get as close to 63 as we like. So we *define* the instantaneous speed to be 63 mph: the value the average speeds approach as the interval shrinks to nothing.

Algebra confirms the table and shows exactly why:

$$\begin{aligned}
\frac{d(1.5+h) - d(1.5)}{h} &= \frac{21(1.5+h)^2 - 21(1.5)^2}{h} \\
&= \frac{21(2.25 + 3h + h^2) - 21(2.25)}{h} && \text{(expand the square)} \\
&= \frac{63h + 21h^2}{h} && \text{(the 47.25s cancel)} \\
&= 63 + 21h && \text{(divide by } h\text{, allowed since } h \neq 0\text{).}
\end{aligned}$$

For every nonzero h, the average speed is $63 + 21h$. As h gets closer to 0, this gets closer to 63. We never set $h = 0$ (the original fraction would be $\frac{0}{0}$); we ask what the expression *approaches*.

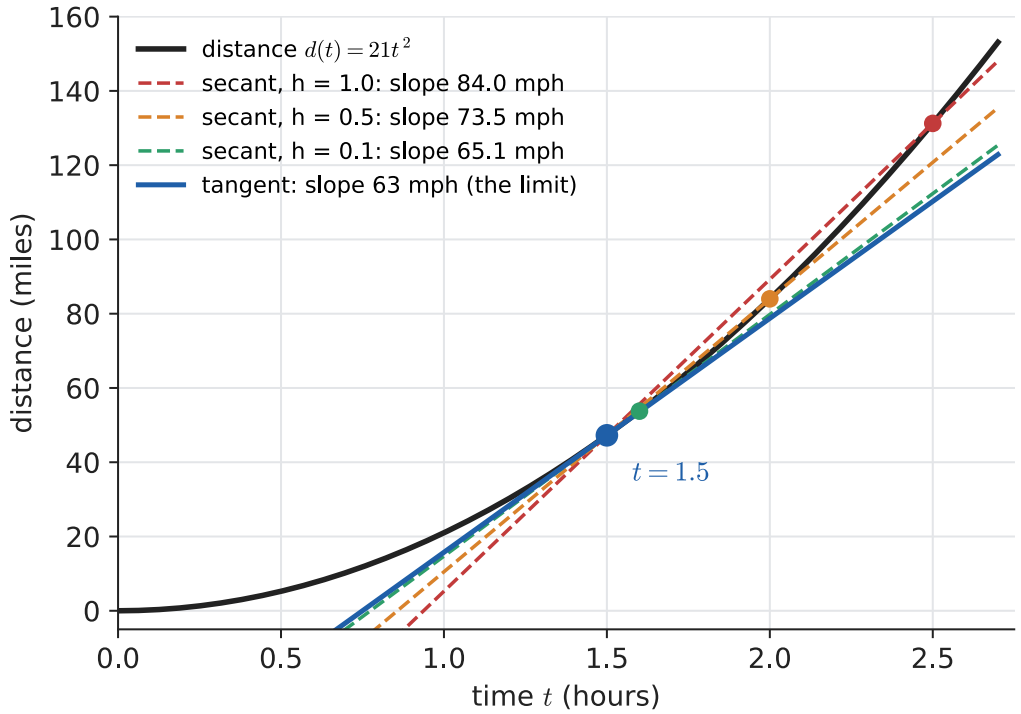


Figure 2. Secant lines over shrinking intervals swing towards the tangent line at t = 1.5, whose slope is the instantaneous speed.

On the distance–time graph, each average speed is the slope of a **secant** line through two points of the curve. As the second point slides towards the first, the secants pivot towards the **tangent** line, which just touches the curve at $t = 1.5$. Its slope is the instantaneous speed. This is the derivative, and it is the subject of Chapter 17. Before we can do that properly, we need to understand the process at its heart: the **limit**.



Secant to tangent

Show the graph of $d(t) = 21t^2$ with a fixed point at $t = 1.5$ and a draggable second point. Display the secant line, its slope, and h. As the user drags the second point in from either side, a table records the slopes, and the secant visibly rotates into the tangent. Let the user change $d(t)$ to $\sqrt{t}$, $\sin t$, or $|t - 1.5|$ (where the left and right secants disagree).

Try it: Try |t − 1.5|: no tangent · Open the secant-to-tangent lab · Slide h: secant to tangent

16.1.3. Where you'd use this

- **Speed cameras (everyday):** UK **average-speed** cameras photograph your number plate at two points and divide distance by time. Covering a 2 km section in 1 minute 20 seconds means $2/(80/3600) = 90$ km/h, even if you briefly hit 110 km/h in the middle. A police radar or laser gun instead measures over a few milliseconds, effectively a very small h. The radar is approximating the limit. The average-speed camera deliberately doesn't.

- **Sport and fitness:** a running watch shows your "current pace" from GPS positions a second or two apart, which is a secant over a short interval. Make the interval too short and GPS jitter (a few metres of error) swamps the real movement, so the reading jumps about. That is the same trade-off between a small h and noise that Section 16.11 finds in floating-point arithmetic.
- **Physics and computing:** every physics engine in a game advances positions using velocities, that is, instantaneous rates, over small time steps Δt (often $1/60$ s). The smaller the step, the closer the simulation is to the limit, and the more it costs to run.

16.2. The idea of a limit

16.2.1. Intuition

The pattern is general. We have a function f and a point a, and we want to know what value $f(x)$ approaches as x gets closer and closer to a — often precisely because we cannot (or do not want to) evaluate $f(a)$ itself.



Limit (informal)

We write

$$\lim_{x \to a} f(x) = L$$

and say "the **limit** of $f(x)$ as x **approaches** (or **tends to**) a is L" if $f(x)$ can be made as close to L as we like by taking x sufficiently close to a, *but not equal to* a. Reading the notation piece by piece: **lim** abbreviates "limit"; the subscript $x \to a$ says which variable is moving and where it is heading; $f(x)$ is the expression being watched; L is the value it approaches. An alternative notation is $f(x) \to L$ **as** $x \to a$.

Two features of the definition are crucial.

- **x never equals a.** The limit is about the behaviour of f *near* a, not *at* a. The value $f(a)$ may be different from the limit, or not exist at all.
- **x approaches from both sides.** Unless stated otherwise, x may approach a from the left (values less than a) or from the right, and $f(x)$ must approach the same L either way.

16.2.2. A function with a hole

Consider *Try it:* Explore the hole at 2 .

$$f(x) = \frac{x^2 - 4}{x - 2}.$$

At $x = 2$ this is $\frac{0}{0}$: undefined. But a table of values near 2 tells a clear story:

x	1.9	1.99	1.999	2.001	2.01	2.1
$f(x)$	3.9	3.99	3.999	4.001	4.01	4.1

From both sides, $f(x)$ approaches 4. The algebra explains it: for $x \neq 2$,

$$\frac{x^2 - 4}{x - 2} = \frac{(x - 2)(x + 2)}{x - 2} = x + 2,$$

so the graph is the line $y = x + 2$ with a single point missing at $(2, 4)$. Therefore $\lim_{x \to 2} f(x) = 4$, even though $f(2)$ does not exist.

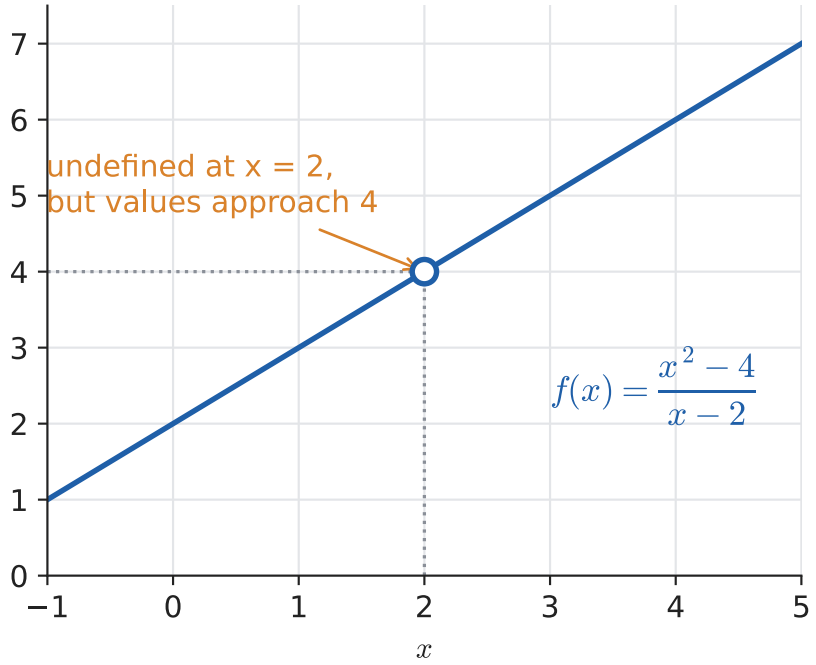


Figure 3. The graph of (x² - 4)/(x - 2) is a straight line with a hole. The limit at 2 is 4, the height the hole "should" have.

This is exactly the situation of the car: the average-speed formula $\frac{d(1.5+h)-d(1.5)}{h}$ is undefined at $h = 0$, but equals $63 + 21h$ everywhere else, so its limit as $h \to 0$ is 63. **Every derivative is a limit of this "0/0" type.** That is why limits come first.



Tables of values can lie

A table suggests a limit; it does not prove one. Consider $g(x) = \sin(\pi/x)$ as $x \to 0$. At $x = 0.1, 0.01, 0.001, \ldots$ we get $\sin(10\pi), \sin(100\pi), \sin(1000\pi), \ldots$, all exactly 0. The table screams "the limit is 0". But at $x = \frac{2}{5}, \frac{2}{9}, \frac{2}{13}, \ldots$ (which also approach 0), $g(x) = \sin(\frac{5\pi}{2}), \sin(\frac{9\pi}{2}), \ldots = 1$. The function oscillates between -1 and 1 infinitely often near 0, and has no limit. Tables sample; algebra and reasoning decide. Use tables to form a conjecture, then confirm it.

Try it: See a table lie .

16.3. One-sided limits, and limits that do not exist

16.3.1. One-sided limits

Sometimes a function behaves differently on the two sides of a point. The **sign function** $\frac{|x|}{x}$ equals -1 for every negative x and $+1$ for every positive x (and is undefined at 0). *Try it:* One-sided limits of |x|/x .



One-sided limits

$\lim_{x \to a^-} f(x) = L$ (the **left-hand limit**) means $f(x)$ approaches L as x approaches a *from below* ($x < a$). $\lim_{x \to a^+} f(x) = L$ (the **right-hand limit**) means the same with $x > a$. The superscript $-$ or $+$ is a direction, not a sign on a: $x \to 2^-$ means x increases towards 2 through values like 1.9, 1.99, …

Key fact: the two-sided limit $\lim_{x \to a} f(x)$ exists and equals L *exactly when* both one-sided limits exist and both equal L.

For the sign function, $\lim_{x \to 0^-} \frac{|x|}{x} = -1$ and $\lim_{x \to 0^+} \frac{|x|}{x} = 1$. They disagree, so $\lim_{x \to 0} \frac{|x|}{x}$ does not exist.

One-sided limits also handle edges of domains: $\sqrt{x}$ is only defined for $x \geq 0$, and $\lim_{x \to 0^+} \sqrt{x} = 0$ is the only sensible question to ask at 0.

16.3.2. Three ways for a limit to fail

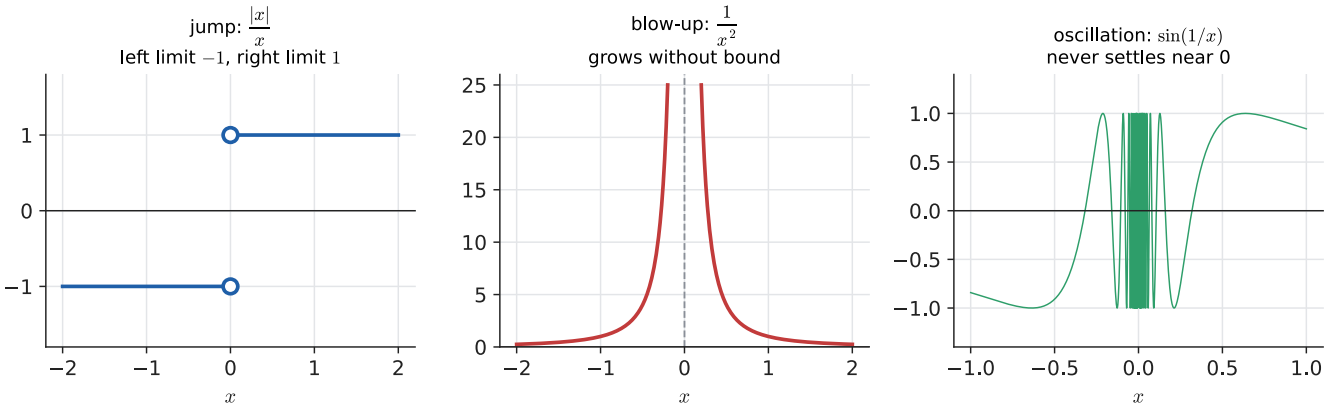


Figure 4. Three typical ways a limit can fail to exist: a jump, unbounded growth, and endless oscillation.

1. **Jump.** The one-sided limits exist but differ, as for $\frac{|x|}{x}$ at 0, or the floor function $\lfloor x \rfloor$ (round down) at any integer. *Try it:* Watch 1/x² blow up ; Watch sin(1/x) oscillate .
2. **Blow-up.** $f(x)$ grows without bound. As $x \to 0$, $\frac{1}{x^2}$ exceeds every number you name. We write $\lim_{x \to 0} \frac{1}{x^2} = \infty$ — but this is a description of *how* the limit fails, not a number. For $\frac{1}{x}$, the two sides go opposite ways: $\lim_{x \to 0^+} \frac{1}{x} = +\infty$, $\lim_{x \to 0^-} \frac{1}{x} = -\infty$.
3. **Oscillation.** $\sin(1/x)$ swings between -1 and 1 infinitely often in every interval around 0, never settling.

Infinite limit

$\lim_{x\to a} f(x) = \infty$ means $f(x)$ becomes larger than any given number when x is close enough to a ($x \neq a$). Similarly for $-\infty$. The limit does **not** exist as a real number; the notation records the direction of failure. The line $x = a$ is then a **vertical asymptote** of the graph.

16.4. Limit laws and direct substitution

16.4.1. The problem

Tables are slow and unreliable. We want rules that compute limits of complicated expressions from limits of simple ones.

16.4.2. The laws

Suppose $\lim_{x\to a} f(x) = L$ and $\lim_{x\to a} g(x) = M$ both exist, and c is a constant. Then:

$$\lim_{x\to a}\big(f(x) \pm g(x)\big) = L \pm M \qquad \textbf{(sum and difference)}$$

$$\lim_{x\to a} c\,f(x) = cL \qquad \textbf{(constant multiple)}$$

$$\lim_{x\to a} f(x)\,g(x) = LM \qquad \textbf{(product)}$$

$$\lim_{x\to a} \frac{f(x)}{g(x)} = \frac{L}{M} \quad \text{provided } M \neq 0 \qquad \textbf{(quotient)}$$

$$\lim_{x\to a} \big(f(x)\big)^n = L^n, \qquad \lim_{x\to a} \sqrt[n]{f(x)} = \sqrt[n]{L} \ \ (L > 0 \text{ if } n \text{ even}) \qquad \textbf{(powers and roots)}$$

together with the two basic limits $\lim_{x\to a} c = c$ and $\lim_{x\to a} x = a$.

These are intuitive — if f is near L and g is near M, then $f + g$ is near $L + M$ — and they can be proved from the formal definition in Section 16.10. The essential condition is in the first line: **both limits must exist.** The quotient law has a second condition: **the denominator's limit must not be zero.** Almost every interesting limit violates that second condition, which is why we need techniques.

16.4.3. Direct substitution

Applying the laws repeatedly to a polynomial gives $\lim_{x\to a} p(x) = p(a)$: just substitute. The same holds for a rational function $\frac{p(x)}{q(x)}$ whenever $q(a) \neq 0$, and, as we will see under continuity, for every "standard" function (roots, exponentials, logarithms, sine, cosine) at points of its domain.

Worked example. $\lim_{x\to 3} \dfrac{x^2 + 2x}{x - 1}$.

$$\lim_{x\to 3} \frac{x^2 + 2x}{x - 1} = \frac{3^2 + 2(3)}{3 - 1} = \frac{15}{2} \qquad \textbf{(denominator } 2 \neq 0\textbf{, so substitution is valid).}$$

Always try substitution first. If it gives a number, you are done. If it gives $\frac{c}{0}$ with $c \neq 0$, the function blows up (examine the signs on each side). If it gives $\frac{0}{0}$, you have an **indeterminate form** — the answer could be anything, and you need to do more work.

Indeterminate form

An **indeterminate form** is a limit whose naive substitution gives an expression like $\frac{0}{0}$, $\frac{\infty}{\infty}$, $\infty - \infty$, $0 \cdot \infty$, or 1^{∞}. "Indeterminate" means the form alone does not determine the answer: $\frac{x}{x}$, $\frac{x^2}{x}$, and $\frac{x}{x^2}$ are all $\frac{0}{0}$ at 0, with limits 1, 0, and "does not exist". The expression must be transformed until substitution works.

16.5. Techniques for 0/0

16.5.1. Factor and cancel

If substituting $x = a$ makes a polynomial numerator and denominator both zero, then by the factor theorem (Volume 1, Chapter 7) both contain the factor $(x - a)$. Factor, cancel it (legitimate because $x \neq a$ inside a limit), then substitute.

Worked example. $\lim_{x \to 3} \dfrac{x^2 - 9}{x^2 - x - 6}$. Substitution gives $\frac{0}{0}$. *Try it:* Check this limit .

$$\begin{aligned} \lim_{x \to 3} \frac{x^2 - 9}{x^2 - x - 6} &= \lim_{x \to 3} \frac{(x-3)(x+3)}{(x-3)(x+2)} && \text{(factor both)} \\ &= \lim_{x \to 3} \frac{x+3}{x+2} && \text{(cancel } x-3\text{; valid since } x \neq 3\text{)} \\ &= \frac{6}{5} && \text{(substitute).} \end{aligned}$$

Worked example. $\lim_{x \to 1} \dfrac{x^3 - 1}{x - 1}$. Use $x^3 - 1 = (x-1)(x^2 + x + 1)$: *Try it:* Check this limit .

$$\lim_{x \to 1} \frac{(x-1)(x^2+x+1)}{x-1} = \lim_{x \to 1} (x^2 + x + 1) = 3.$$

16.5.2. Rationalise

When square roots are involved, multiply by the **conjugate** (the same expression with the sign between the terms flipped) to use $(A - B)(A + B) = A^2 - B^2$ and clear the root.

Worked example. $\lim_{x \to 0} \dfrac{\sqrt{x+9} - 3}{x}$. Substitution: $\frac{3-3}{0} = \frac{0}{0}$. *Try it:* Check this limit .

$$\begin{aligned} \lim_{x \to 0} \frac{\sqrt{x+9} - 3}{x} &= \lim_{x \to 0} \frac{(\sqrt{x+9} - 3)(\sqrt{x+9} + 3)}{x(\sqrt{x+9} + 3)} && \text{(multiply top and bottom by the conjugate)} \\ &= \lim_{x \to 0} \frac{(x+9) - 9}{x(\sqrt{x+9} + 3)} && \left((A-B)(A+B) = A^2 - B^2\right) \\ &= \lim_{x \to 0} \frac{1}{\sqrt{x+9} + 3} && \text{(cancel } x\text{)} \\ &= \frac{1}{3+3} = \frac{1}{6}. \end{aligned}$$

A table agrees: at $x = 0.001$, $\frac{\sqrt{9.001} - 3}{0.001} \approx 0.166662$.

16.5.3. Combine fractions

Compound fractions are simplified by finding a common denominator.

Worked example. $\displaystyle\lim_{x\to 0}\frac{\frac{1}{x+2}-\frac{1}{2}}{x}$.

$$\frac{\frac{1}{x+2}-\frac{1}{2}}{x}=\frac{\frac{2-(x+2)}{2(x+2)}}{x} \qquad \text{(common denominator } 2(x+2)\text{)}$$

$$=\frac{-x}{2x(x+2)}=\frac{-1}{2(x+2)} \qquad \text{(cancel } x\text{)},$$

so the limit is $\frac{-1}{2(0+2)}=-\frac{1}{4}$.

16.5.4. The method

Example 2. Procedure — evaluate lim f(x) as x → a

Input: an expression $f(x)$ and a point a.

Steps:

1. **Substitute** $x=a$. If you get a real number (with no division by zero), that is the limit. Stop.
2. If you get $\frac{c}{0}$ with $c\neq 0$: the limit is infinite or does not exist. Check the sign of the expression on each side of a to decide between $+\infty$, $-\infty$, or "different on each side". Stop.
3. If you get $\frac{0}{0}$, **transform** the expression:
   a. polynomials → factor and cancel the common factor $(x-a)$;
   b. square roots → multiply by the conjugate;
   c. compound fractions → combine over a common denominator;
   d. trigonometric functions → rewrite in terms of $\frac{\sin u}{u}$ (next section).
4. Substitute again into the simplified expression; repeat step 3 if still $\frac{0}{0}$.
5. If nothing works, examine a table and graph to form a conjecture, then look for a squeeze argument (next section) or use L'Hôpital's rule (Chapter 18).
6. Where the function is defined piecewise or involves $|x|$, compute the one-sided limits separately.

Output: the limit, or a statement of how it fails to exist.

Choosing a technique

The structure of the expression chooses for you. Factor-and-cancel is the default for rational functions; conjugates are for differences involving square roots; common denominators are for fractions within fractions. All three do the same job: *expose and cancel the factor that is making both top and bottom vanish.* Numerical tables are for checking, never for proving. L'Hôpital's rule (using derivatives) is a powerful general method you will meet later, but for most algebraic limits the techniques here are faster and give more insight.



You cannot cancel at the point itself

Cancelling $(x-2)$ from $\frac{(x-2)(x+2)}{x-2}$ is valid only because $x \neq 2$. The functions $\frac{x^2-4}{x-2}$ and $x+2$ are *different functions* (one is undefined at 2), but they agree everywhere except at 2, so they have the same limit there. A limit never looks at the point itself — which is precisely what makes the cancellation legitimate.

US curriculum

Limits are previewed informally in **Precalculus** and form Unit 1 ("Limits and Continuity") of **AP Calculus AB/BC**. Students are expected to estimate limits from tables and graphs, compute them algebraically (factoring, rationalising), and evaluate one-sided limits. The ε–δ definition is usually optional in high school and appears in college calculus or a first course in **real analysis**.

16.6. The squeeze theorem and sin x / x

16.6.1. The problem

Some limits resist algebra entirely. The most important one is

$$\lim_{x \to 0} \frac{\sin x}{x}.$$

Substitution gives $\frac{0}{0}$, and there is nothing to factor. Yet this limit is the key to the derivative of sine (and so to all of oscillations, waves, and signal processing). A table, with x in radians, suggests the answer:

```python
import math

def f(x):
    return math.sin(x) / x

for x in [1, 0.5, 0.1, 0.01, 0.001, -0.001, -0.1]:
    print(f"{x:>7}  {f(x):.10f}")
```

```
     1  0.8414709848
   0.5  0.9588510772
   0.1  0.9983341665
  0.01  0.9999833334
 0.001  0.9999998333
-0.001  0.9999998333
  -0.1  0.9983341665
```

It looks like 1. To prove it we need a new tool.

16.6.2. The squeeze theorem



Squeeze theorem

Squeeze theorem (also **sandwich** or **pinching theorem**): if $g(x) \le f(x) \le h(x)$ for all x near a (except possibly at a), and $\lim_{x\to a} g(x) = \lim_{x\to a} h(x) = L$, then $\lim_{x\to a} f(x) = L$ as well.

The intuition is two police officers escorting a suspect: if both officers walk into the station, so does the suspect between them.

Warm-up example. $\lim_{x\to 0} x^2 \sin(1/x)$. We know nothing useful about $\sin(1/x)$ near 0 — it oscillates wildly — except that it stays between -1 and 1. Multiplying by $x^2 \ge 0$: *Try it:* Squeeze x² sin(1/x) to 0 .

$$-x^2 \le x^2 \sin(1/x) \le x^2.$$

Both outer functions tend to 0, so $\lim_{x\to 0} x^2 \sin(1/x) = 0$. The wild oscillation is crushed by the shrinking factor.

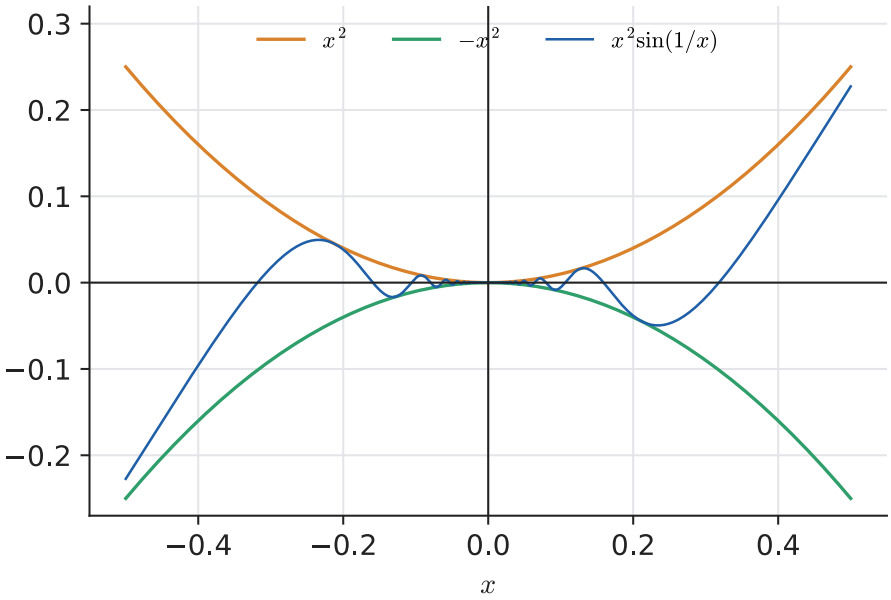


Figure 5. x² sin(1/x) oscillates infinitely often near 0, but is trapped between -x² and x², which squeeze it to 0.

16.6.3. Proof that sin x / x → 1

Take $0 < x < \frac{\pi}{2}$ and draw an angle x (in radians) at the centre O of a unit circle. Let $A = (1, 0)$, let $B = (\cos x, \sin x)$ be the point on the circle, and let T be the point where the line OB meets the vertical tangent line at A; by similar triangles $AT = \tan x$. *Try it:* Tabulate sin x / x .

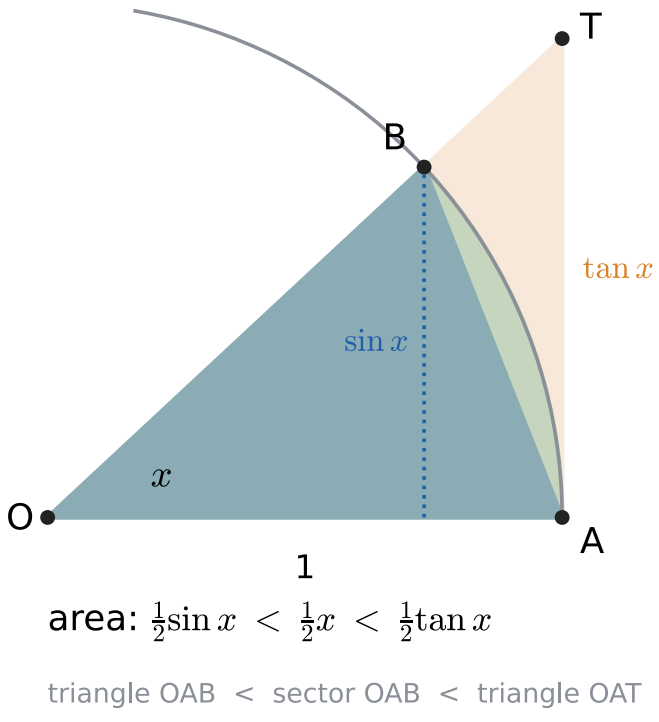


Figure 6. Three nested regions: triangle OAB, sector OAB, and triangle OAT. Comparing their areas traps sin x / x.

The three regions are nested, so their areas are in order:

$$\underbrace{\tfrac{1}{2}\cdot 1\cdot \sin x}_{\text{triangle } OAB} \;<\; \underbrace{\tfrac{1}{2}\cdot 1^2\cdot x}_{\text{sector } OAB} \;<\; \underbrace{\tfrac{1}{2}\cdot 1\cdot \tan x}_{\text{triangle } OAT}.$$

(The triangle OAB has base 1 and height $\sin x$; the sector has area $\tfrac{1}{2}r^2\theta$ with $r = 1$, which is where radians are essential; triangle OAT has base 1 and height $\tan x$.) Now manipulate:

$$\begin{aligned} \sin x < x < \tan x = \frac{\sin x}{\cos x} \qquad & \text{(multiply by 2)} \\ 1 < \frac{x}{\sin x} < \frac{1}{\cos x} \qquad & \text{(divide by } \sin x > 0\text{)} \\ \cos x < \frac{\sin x}{x} < 1 \qquad & \text{(take reciprocals, which reverses the inequalities).} \end{aligned}$$

As $x \to 0^+$, $\cos x \to 1$. The squeeze theorem gives $\lim_{x\to 0^+} \frac{\sin x}{x} = 1$. For negative x, note that $\frac{\sin(-x)}{-x} = \frac{-\sin x}{-x} = \frac{\sin x}{x}$ (the function is **even**), so the left-hand limit is the same. Therefore

$$\boxed{\lim_{x\to 0} \frac{\sin x}{x} = 1.}$$

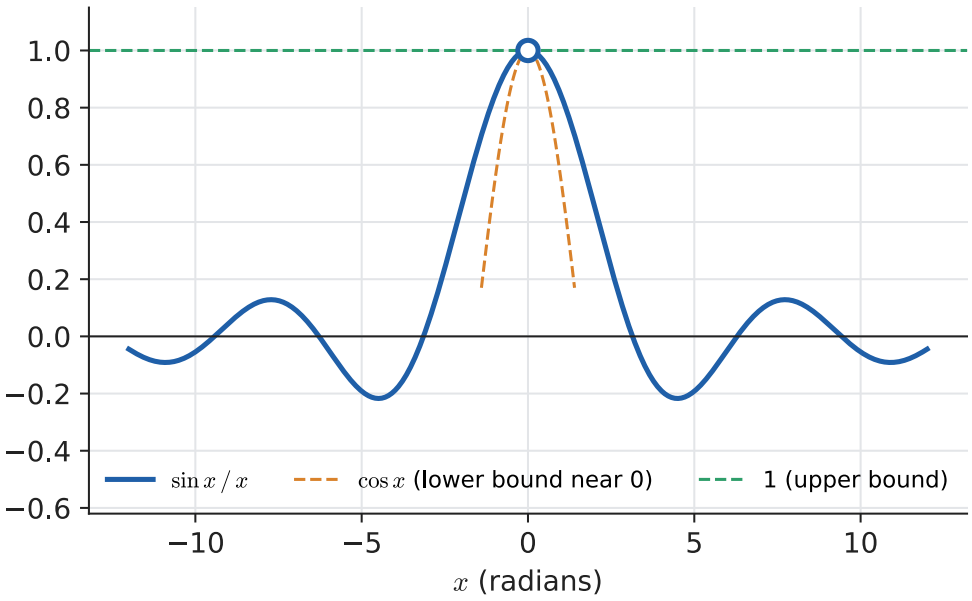


Figure 7. sin x / x is undefined at 0 but approaches 1, squeezed between cos x and 1.

Geometrically: for small angles, the arc length x and the half-chord $\sin x$ are almost the same. This is the **small-angle approximation** $\sin x \approx x$, used throughout physics and engineering (the simple pendulum, optics, surveying).

16.6.4. Where you'd use this

The small-angle approximation $\sin x \approx \tan x \approx x$ (radians) turns angles into lengths with no trigonometry.

- **Outdoors and sport:** rifle scopes and some binoculars are marked in **milliradians** (thousandths of a radian). An object of height H metres that spans m milliradians is about $1000H/m$ metres away, since $H/d \approx \tan(m/1000) \approx m/1000$. A 1.8 m person spanning 3 milliradians is about

$1000 \times 1.8/3 = 600$ m away.

- **Astronomy:** the Moon spans about 0.52°, which is $0.52 \times \pi/180 \approx 0.0091$ rad. At a distance of 384,400 km its diameter is about $384{,}400 \times 0.0091 \approx 3{,}490$ km (the true value is 3,474 km). Telescope and camera specifications ("arcseconds per pixel") are all **size ≈ distance × angle**.
- **Computing:** the function $\operatorname{sinc} x = \sin x / x$ (with value 1 at 0, using this limit) is the ideal interpolation filter in signal processing. Image-resizing filters such as **Lanczos** (an option in most image editors and in libraries like Pillow) are built from it, and so is audio sample-rate conversion. The code has to special-case $x = 0$, and it does that by returning the limit, 1.

16.6.5. Using the result

Worked example. $\lim_{x \to 0} \dfrac{\sin 3x}{x}$. Substitute $u = 3x$ (so $u \to 0$ as $x \to 0$): *Try it:* Check this limit .

$$\lim_{x \to 0} \frac{\sin 3x}{x} = \lim_{x \to 0} 3 \cdot \frac{\sin 3x}{3x} = 3 \lim_{u \to 0} \frac{\sin u}{u} = 3 \cdot 1 = 3.$$

Worked example. $\lim_{x \to 0} \dfrac{1 - \cos x}{x}$. Multiply by the "conjugate" $1 + \cos x$: *Try it:* Check this limit .

$$\begin{aligned} \frac{1 - \cos x}{x} &= \frac{(1 - \cos x)(1 + \cos x)}{x(1 + \cos x)} = \frac{1 - \cos^2 x}{x(1 + \cos x)} \\ &= \frac{\sin^2 x}{x(1 + \cos x)} && (\sin^2 + \cos^2 = 1) \\ &= \frac{\sin x}{x} \cdot \frac{\sin x}{1 + \cos x} \;\to\; 1 \cdot \frac{0}{2} = 0. \end{aligned}$$

These two limits are exactly what is needed in Chapter 17 to show that the derivative of $\sin x$ is $\cos x$.



Degrees ruin everything

The result $\frac{\sin x}{x} \to 1$ holds only when x is in **radians**. In degrees, $\sin(x^\circ) = \sin\left(\frac{\pi x}{180}\right)$, so $\frac{\sin(x^\circ)}{x} \to \frac{\pi}{180} \approx 0.01745$. The area formula $\frac{1}{2} r^2 \theta$ for a sector also assumes radians. This is the deep reason calculus always uses radians: they are the unit in which $\sin x \approx x$ for small x. It is also a classic bug: `math.sin` expects radians.

16.7. Continuity

16.7.1. The problem

For most functions you meet, the limit at a point is just the value at the point: $\lim_{x \to 2} x^2 = 4 = 2^2$. The interesting cases are where this fails. We want a name for the good behaviour, and a catalogue of the ways it can break.

16.7.2. Intuition

A function is continuous if you can draw its graph without lifting your pen: no holes, no jumps, no vertical asymptotes, no wild oscillations. Small changes in input produce small changes in output. Physical quantities — the temperature in a room over time, the position of a car — are continuous; a step change in price at a tax threshold is not.



Continuity at a point

A function f is **continuous at** a if all three conditions hold:

1. $f(a)$ is defined (a is in the domain);
2. $\lim_{x \to a} f(x)$ exists;
3. the two are equal: $\lim_{x \to a} f(x) = f(a)$.

f is **continuous on an interval** if it is continuous at every point of it (using one-sided limits at closed endpoints), and simply **continuous** if it is continuous at every point of its domain. A point where continuity fails is a **discontinuity**.

Condition 3 is the heart of it: *the limit is what you get by substituting.* That is why direct substitution works — it works precisely for continuous functions.

16.7.3. Types of discontinuity

Each condition can fail, giving a catalogue that maps onto the pictures earlier in the chapter:

- **Removable discontinuity** (a hole): the limit exists but $f(a)$ is undefined or different. $\frac{x^2-4}{x-2}$ at 2. "Removable" because redefining $f(2) = 4$ repairs it. Similarly, $\frac{\sin x}{x}$ has a removable discontinuity at 0; the repaired function (defined to be 1 at 0) is called $\operatorname{sinc} x$ and is important in signal processing.
- **Jump discontinuity**: both one-sided limits exist but differ. The floor function at each integer; postage rates; tax brackets.
- **Infinite discontinuity**: the function blows up — a vertical asymptote. $\frac{1}{x}$ at 0, $\tan x$ at $\frac{\pi}{2}$.
- **Oscillating discontinuity**: no one-sided limit exists. $\sin(1/x)$ at 0.

16.7.4. Which functions are continuous?

A great deal follows from the limit laws:

- Polynomials are continuous everywhere.
- Rational functions are continuous everywhere *in their domain* (i.e. except where the denominator is zero).
- $\sqrt[n]{x}$, e^x, $\ln x$, $\sin x$, $\cos x$, $|x|$ are continuous on their domains.
- Sums, differences, products, quotients (where the denominator is nonzero), and **compositions** of continuous functions are continuous.

So $e^{\sin x} + \sqrt{x^2+1}$ is continuous everywhere, with no work. The composition rule also lets limits pass inside continuous functions: $\lim_{x\to a} g(f(x)) = g\left(\lim_{x\to a} f(x)\right)$ when g is continuous at the inner limit.

Worked example (making a piecewise function continuous). For which value of k is

$$f(x) = \begin{cases} kx + 1, & x \le 2 \\ x^2 - 1, & x > 2 \end{cases}$$

continuous everywhere? Each piece is a polynomial, hence continuous; the only danger is the join at $x = 2$.

1. $f(2) = 2k + 1$ (the first piece includes $x = 2$).
2. Left limit: $\lim_{x\to 2^-}(kx + 1) = 2k + 1$. Right limit: $\lim_{x\to 2^+}(x^2 - 1) = 3$. The limit exists iff $2k + 1 = 3$.
3. Then it equals $f(2)$ automatically.

So $k = 1$.



> *Continuity is about the domain*
> Is $\frac{1}{x}$ continuous? By the definition, yes: it is continuous at every point *of its domain* (all $x \neq 0$). Its graph has two pieces, but 0 is not in the domain, so there is no point at which continuity fails. Some textbooks nevertheless say "$\frac{1}{x}$ has a discontinuity at 0"; be aware that the phrase is used loosely. What matters for theorems like the IVT below is continuity on a whole *interval*, and $\frac{1}{x}$ is not continuous on any interval containing 0.

16.7.5. Where you'd use this

Continuity is the promise that **a small change in input gives a small change in output**. Discontinuities are where that promise breaks, and they often matter more than the smooth parts.

- **Shopping:** an online shop charges \$4.99 delivery below \$35 and nothing at \$35 or more. The total you pay has a jump discontinuity at \$35. A \$33 basket costs \$37.99, but a \$35 basket costs \$35.00: adding a \$2 item gets you more and you pay \$2.99 **less**. Tax bands, parcel weight limits and phone-data caps all create jumps like this, and people (rationally) bunch up just past them.
- **Engineering:** in a lift, a robot arm or a roller coaster, a jump in velocity means an impossible infinite acceleration, and in practice a violent jolt. Motion controllers plan paths whose position, velocity and acceleration are all continuous, so that the ride is smooth.
- **Computing:** graphics and animation curves (Bézier curves, splines) are joined so that position and direction are continuous where pieces meet. Otherwise you see a kink or a "pop" between frames. In code, a jump discontinuity is usually an `if` statement, such as `if total >= 35:`, and the boundary value is exactly where off-by-one and floating-point bugs live: is 34.999999999 less than 35?

16.8. The intermediate value theorem and bisection

16.8.1. The problem

Does $x^3 - x - 2 = 0$ have a solution? It is a cubic with no rational roots (the candidates $\pm 1, \pm 2$ all fail), so factoring will not help. But $f(1) = 1 - 1 - 2 = -2 < 0$ and $f(2) = 8 - 2 - 2 = 4 > 0$. The graph of a continuous function cannot get from below the axis to above it without crossing. That intuition is a theorem.



Intermediate value theorem

Intermediate value theorem (IVT): if f is continuous on the closed interval $[a, b]$ and N is any number between $f(a)$ and $f(b)$, then there is at least one c in $[a, b]$ with $f(c) = N$.

The most-used special case: if f is continuous on $[a, b]$ and $f(a)$ and $f(b)$ have **opposite signs**, then f has a **root** (a zero) somewhere in (a, b).

The IVT tells you a root *exists*; it does not tell you where. It also needs continuity: $f(x) = \frac{1}{x}$ has $f(-1) = -1 < 0 < 1 = f(1)$ but no root, because it is not continuous on $[-1, 1]$.

16.8.2. From theorem to algorithm: bisection

The IVT becomes an algorithm with one observation. If there is a sign change on $[a, b]$, evaluate f at the midpoint m. Either $f(m) = 0$ (done), or the sign change is in $[a, m]$, or it is in $[m, b]$. Keep the half with the sign change and repeat. The interval known to contain a root halves every step — it is binary search on the real line.

Example 3. Procedure — bisection method

Input: a continuous function f; an interval $[a, b]$ with $f(a)$ and $f(b)$ of opposite signs; a tolerance $\varepsilon > 0$.

Steps:

1. Compute the midpoint $m = \frac{a+b}{2}$ and $f(m)$.
2. If $f(m) = 0$ or the half-width $\frac{b-a}{2} < \varepsilon$, return m.
3. If $f(a)$ and $f(m)$ have opposite signs, set $b \leftarrow m$; otherwise set $a \leftarrow m$.
4. Go to step 1.

Output: m, within ε of a true root. After n halvings the interval has width $\frac{b-a}{2^n}$, so about $\log_2 \frac{b-a}{\varepsilon}$ steps are needed.

Worked example by hand. $f(x) = x^3 - x - 2$ on $[1, 2]$. *Try it:* Step through this table .

step	a	b	midpoint m	$f(m)$	root is in
1	1	2	1.5	-0.125	$[1.5, 2]$
2	1.5	2	1.75	1.609	$[1.5, 1.75]$
3	1.5	1.75	1.625	0.666	$[1.5, 1.625]$
4	1.5	1.625	1.5625	0.252	$[1.5, 1.5625]$
5	1.5	1.5625	1.53125	0.0591	$[1.5, 1.53125]$
6	1.5	1.53125	1.515625	-0.0341	$[1.515625, 1.53125]$
7	1.515625	1.53125	1.5234375	0.0123	$[1.515625, 1.5234375]$

(Step 1 in detail: $f(1.5) = 3.375 - 1.5 - 2 = -0.125$. Since $f(1.5) < 0 < f(2)$, the root lies in $[1.5, 2]$.) After 7 steps the root is pinned inside an interval of width $2^{-7} \approx 0.0078$: it is about 1.52.

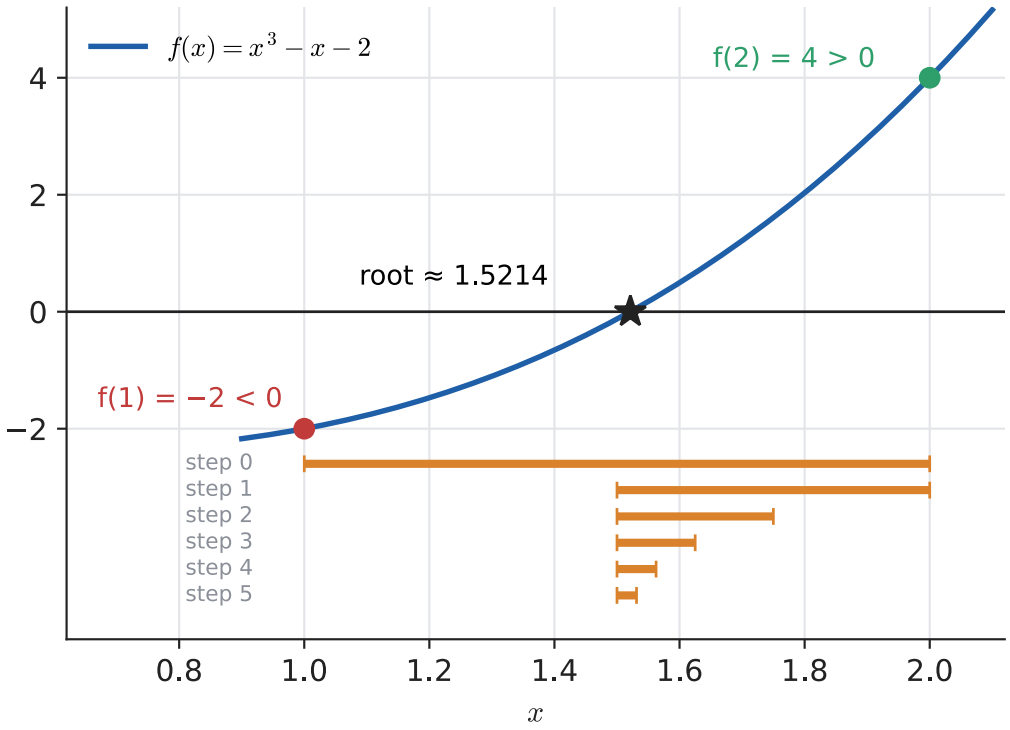


Figure 8. Each bisection step halves the interval known to contain the root.

PYTHON

```python
def bisection(f, a, b, tol=1e-10, max_iter=200):
    """Find a root of continuous f in [a, b], given f(a), f(b) of opposite sign."""
    fa, fb = f(a), f(b)
    if fa * fb > 0:
        raise ValueError("f(a) and f(b) must have opposite signs")
    for n in range(max_iter):
        m = (a + b) / 2
        fm = f(m)
        if fm == 0 or (b - a) / 2 < tol:
            return m, n
        if fa * fm < 0:          # sign change in left half
            b, fb = m, fm
        else:                    # sign change in right half
            a, fa = m, fm
    return (a + b) / 2, max_iter

root, steps = bisection(lambda x: x**3 - x - 2, 1, 2)
print(root, steps)
print(root**3 - root - 2)
```

```
1.5213797068572603 33
3.131943593359665e-10
```

33 halvings shrink the interval from width 1 to about 10^{-10}, as predicted: $\log_2 10^{10} \approx 33.2$.

16.8.3. Choosing a root-finding method

- **Bisection** is slow (one binary digit per step) but **guaranteed**: given a continuous function and a sign change, it cannot fail, and you know the error bound in advance.

- **Newton's method** (Chapter 18, Chapter 32) is far faster near a root — the number of correct digits roughly doubles each step — but needs the derivative and can diverge or wander off from a poor starting point.
- In practice, robust library routines (such as Brent's method, behind `scipy.optimize.brentq`) combine both: fast steps when they behave, bisection as a safety net.

16.8.4. Where you'd use this

- **Finance:** you invest $1,000 and get $600 back after one year and another $600 after two. What annual rate of return is that? It's the rate r that makes the **net present value** zero:

$$f(r) = -1000 + \frac{600}{1+r} + \frac{600}{(1+r)^2} = 0.$$

  $f(0) = 200 > 0$ and $f(1) = -550 < 0$, and f is continuous for $r > -1$, so by the IVT a root lies in between. Bisection finds $r \approx 0.1307$, a 13.07% annual return. A spreadsheet's `IRR` function solves exactly this equation with an iterative root-finder. (Here a quadratic in $1/(1+r)$ would also work, but cash flows over 30 years give a degree-30 polynomial, and then iteration is the only practical option.)
- **Engineering:** "at what speed does this beam start to resonate?", "what pipe diameter gives exactly this flow?" When an equation has no algebraic solution but you can evaluate it and find a sign change, bisection gives a guaranteed answer with a known error bound. Safety-critical code values that guarantee.
- **Computing:** "binary search on the answer". What is the largest batch size that fits in GPU memory? Try 512 (fits) and 1,024 (out of memory), then 768, and so on. Nine more tries pin down the exact limit among the 512 sizes in between. That's bisection on a function that changes from "fits" to "doesn't fit".



When bisection fails

- **No sign change:** x^2 has a root at 0 but is never negative, so bisection cannot even start. Roots where the graph touches the axis without crossing (**double roots**) are invisible to it.
- **Discontinuity:** on $[-1, 1]$, $\frac{1}{x}$ changes sign, and bisection will dutifully converge on $x = 0$ — which is not a root but a vertical asymptote. Bisection trusts the continuity hypothesis; check it.
- **Several roots:** it finds one of them, with no control over which.
- **Underflow in the test:** checking `f(a) * f(m) < 0` can underflow to 0 for tiny values; production code compares signs instead.

The IVT in the world

The IVT has surprising everyday consequences. **The wobbly table:** on any continuous floor (no steps), a square four-legged table can be rotated so all four legs touch the ground — apply the IVT to the height difference of opposite leg pairs as you rotate through $90°$. **Antipodal temperatures:** at any moment there are two diametrically opposite points on the equator with exactly the same temperature (apply the IVT to $T(\theta) - T(\theta + 180°)$, which changes sign over half a turn). And **git bisect**, which binary-searches the commit history for the first bad commit, relies on a discrete version of the same idea: there is a good commit and a bad commit, so somewhere in between the state must switch.

Bisection stepper

Enter a function and an interval. Each click on "step" shades the current interval, marks the midpoint with the sign of $f(m)$, and keeps the half containing the sign change. A side panel plots the interval width on a logarithmic scale (a straight line: halving each step). Include a mode with $1/x$ to watch bisection "converge" to an asymptote.

Try it: Bisection on 1/x · Open the bisection stepper

16.9. Limits at infinity and asymptotes

16.9.1. The problem

What happens to $f(x)$ as x grows without bound? This is the question of **long-run behaviour**: the steady state of a cooling cup of coffee, the terminal velocity of a falling object, the cost of an algorithm on enormous inputs.

Limit at infinity

$\lim_{x \to \infty} f(x) = L$ means $f(x)$ can be made as close to L as we like by taking x sufficiently large. Similarly for $x \to -\infty$. If such an L exists, the horizontal line $y = L$ is a **horizontal asymptote**. For a sequence (Volume 1, Chapter 14), $\lim_{n \to \infty} a_n = L$ means the same thing with n restricted to whole numbers; this is exactly the "convergence" of the previous chapter, now with a name.

The basic facts: $\lim_{x \to \infty} \frac{1}{x} = 0$, and more generally $\lim_{x \to \infty} \frac{1}{x^p} = 0$ for any $p > 0$. Everything else is reduced to these.

16.9.2. Dividing by the highest power

A rational function at infinity is a $\frac{\infty}{\infty}$ indeterminate form. The technique: **divide numerator and denominator by the highest power of x in the denominator**, so every term becomes a constant or tends to 0.

Worked example. $\lim_{x\to\infty} \dfrac{3x^2-5x+1}{2x^2+7}$. *Try it:* Limit at infinity, tabulated .

$$\lim_{x\to\infty} \frac{3x^2-5x+1}{2x^2+7} = \lim_{x\to\infty} \frac{3-\frac{5}{x}+\frac{1}{x^2}}{2+\frac{7}{x^2}} \qquad \text{(divide every term by } x^2\text{)}$$
$$= \frac{3-0+0}{2+0} = \frac{3}{2} \qquad \left(\tfrac{1}{x}, \tfrac{1}{x^2} \to 0\right).$$

The lesson: at infinity, only the **leading terms** matter. $3x^2-5x+1$ "behaves like" $3x^2$ and $2x^2+7$ like $2x^2$.

Example 4. Procedure — limit of a rational function as $x \to \pm\infty$

Input: $\dfrac{p(x)}{q(x)}$ with $\deg p = m$, $\deg q = n$, leading coefficients a_m, b_n.

Steps: compare degrees.

1. $m < n$: the limit is 0 (horizontal asymptote $y = 0$).
2. $m = n$: the limit is $\dfrac{a_m}{b_n}$, the ratio of leading coefficients.
3. $m > n$: no finite limit; the function grows like $\frac{a_m}{b_n}x^{m-n}$, heading to $\pm\infty$. (If $m = n+1$ the graph approaches a slanted line, an **oblique asymptote**, found by polynomial division.)

To justify any case, divide top and bottom by x^n.

Output: the limit, and the horizontal asymptote if there is one.

Finding vertical asymptotes. These occur where the denominator is zero but the numerator is not (after cancelling common factors — a common factor gives a hole instead). For

$$f(x) = \frac{2x^2+1}{x^2-1} = \frac{2x^2+1}{(x-1)(x+1)},$$

the numerator is never zero, so $x = 1$ and $x = -1$ are vertical asymptotes; the degrees are equal, so $y = \frac{2}{1} = 2$ is the horizontal asymptote.

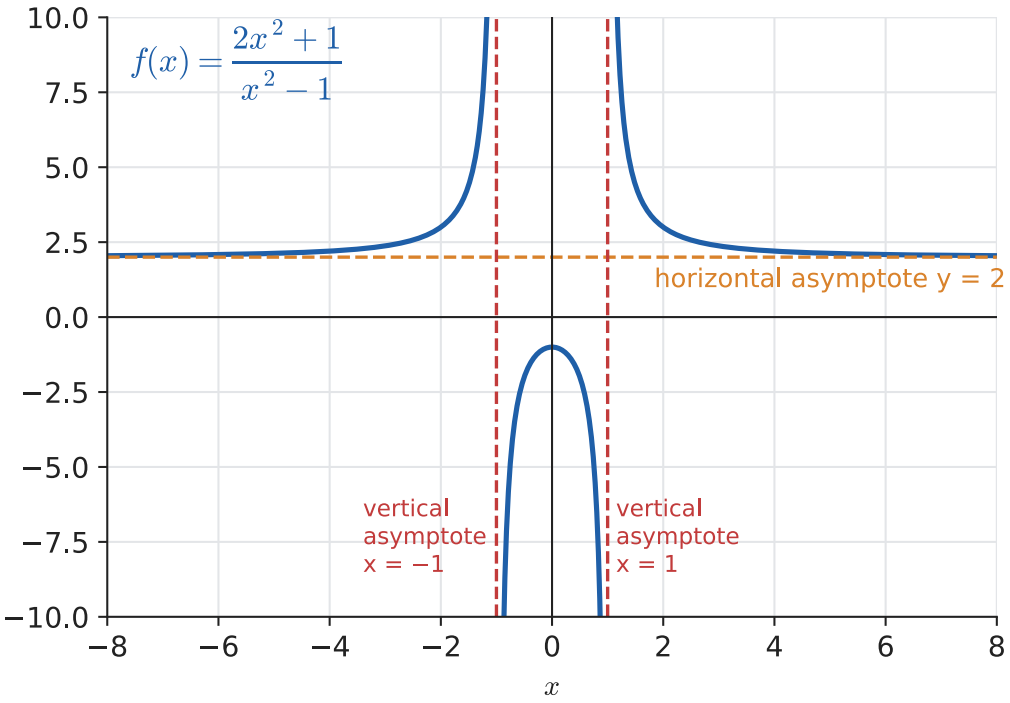


Figure 9. Vertical asymptotes where the denominator vanishes; a horizontal asymptote set by the ratio of leading coefficients.

Worked example (∞ − ∞). $\lim_{x\to\infty}\left(\sqrt{x^2+x}-x\right)$. Both terms grow without bound; their difference is indeterminate. Rationalise: *Try it:* Tabulate √(x²+x) − x .

$$\begin{aligned}\sqrt{x^2+x}-x &= \frac{(\sqrt{x^2+x}-x)(\sqrt{x^2+x}+x)}{\sqrt{x^2+x}+x} = \frac{x^2+x-x^2}{\sqrt{x^2+x}+x}\\ &= \frac{x}{\sqrt{x^2+x}+x} = \frac{1}{\sqrt{1+\frac{1}{x}}+1} \qquad \text{(divide top and bottom by } x>0)\\ &\to \frac{1}{\sqrt{1}+1} = \frac{1}{2}.\end{aligned}$$

16.9.3. The growth hierarchy

Beyond polynomials, the essential facts for limits at infinity (proved with tools from Chapter 18) are that, for any constants $p>0$ and $b>1$,

$$\lim_{x\to\infty}\frac{\ln x}{x^p}=0, \qquad \lim_{x\to\infty}\frac{x^p}{b^x}=0.$$

In words: **logarithms grow slower than any power, and powers grow slower than any exponential.** $\ln x$ loses to $x^{0.01}$ eventually (though "eventually" can be astronomically far out), and x^{100} loses to 1.01^x.

16.9.4. Where you'd use this

Limits at infinity describe **where things settle**, which is often the only question that matters in the long run.

- **Kitchen physics:** a cup of coffee at 90°C in a 20°C room cools according to Newton's law of cooling, roughly $T(t) = 20 + 70e^{-0.05t}$ (t in minutes). After 30 minutes it is $20 + 70e^{-1.5} \approx 35.6$°C. As $t \to \infty$, $e^{-0.05t} \to 0$, so $T \to 20$: the room temperature is a horizontal asymptote. Solving $20 + 70e^{-0.05t} = 25$ gives $t = 20 \ln 14 \approx 52.8$ minutes to get within 5°C of it.

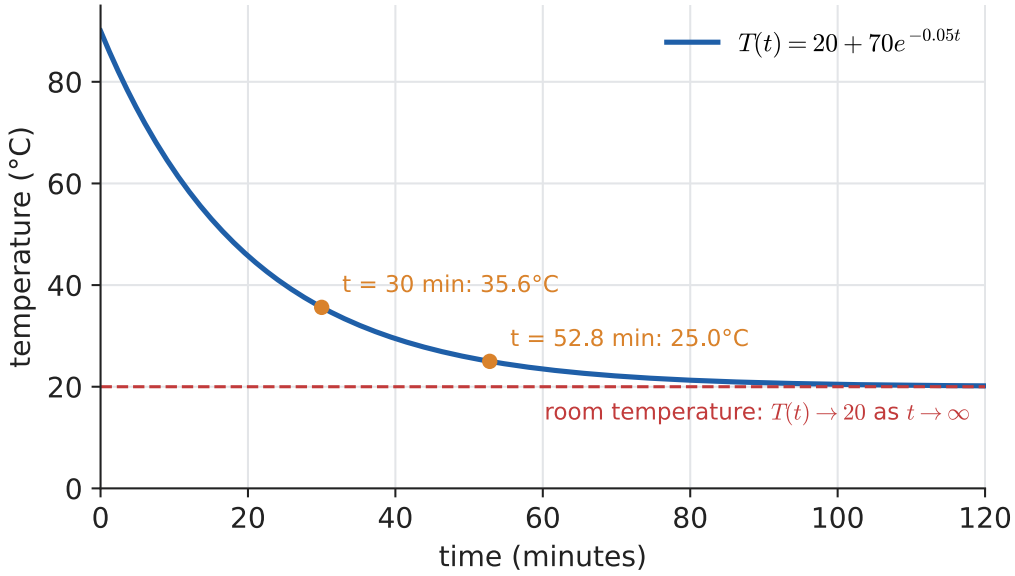


Figure 10. Newton's law of cooling. The temperature approaches the room's 20°C (dashed) but never quite reaches it: a horizontal asymptote.

- **Science:** a skydiver's speed approaches a **terminal velocity** (about 55 m/s, roughly 200 km/h, belly-down) as air resistance grows to match gravity. A charging capacitor's voltage approaches the supply voltage, and a drug level approaches the steady state from Volume 1, Chapter 14 . Each model is "a constant plus a decaying exponential", and the limit is the constant.
- **Business and cloud computing:** a service costs $500 a month in fixed fees plus $0.0004 per request. The cost per request is $\frac{500 + 0.0004n}{n} = \frac{500}{n} + 0.0004$. At a million requests a month that's $0.0009; at a hundred million it's $0.000405. As $n \to \infty$ it tends to $0.0004: fixed costs are spread ever thinner, and only the marginal cost survives. This is "divide by the highest power" again, and it is the arithmetic behind economies of scale.

16.9.5. A famous limit at infinity

Compound interest, compounded ever more frequently, leads to

$$\lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n = e \approx 2.718281828,$$

a 1^∞ indeterminate form: the base tends to 1 but the exponent grows. This is one standard definition of e (Volume 1, Chapter 12). We will come back to it below, because a computer gets it spectacularly wrong.

16.10. Optional depth: the formal ε–δ definition



This section can be skipped on a first reading; nothing later depends on fluency with it. Read it if you want to know what "as close as we like" and "sufficiently close" mean *precisely* — the step that turned calculus from a brilliant collection of techniques into rigorous mathematics.

16.10.1. Why "close" needs a definition

For 150 years after Newton and Leibniz, calculus worked spectacularly while its foundations were shaky: quantities were "infinitely small" but not zero, vanishing but not vanished. Bishop Berkeley mocked these as "the ghosts of departed quantities". The resolution, by Cauchy, Bolzano, and above all Weierstrass in the 19th century, was to eliminate the vague words "approaches" and "infinitely small" entirely, replacing them with statements about ordinary inequalities.

16.10.2. The definition as a game

Think of proving $\lim_{x \to a} f(x) = L$ as a game between a **challenger** and a **prover**.

1. The challenger picks a tolerance $\varepsilon > 0$ (Greek epsilon, for "error"): "I want $f(x)$ within ε of L."
2. The prover must respond with a distance $\delta > 0$ (Greek delta, for "distance"): "Then keep x within δ of a (but not equal to it)."
3. The prover wins the round if *every* such x gives $f(x)$ within ε of L.

The limit is L if the prover can win **every** round, no matter how small an ε the challenger chooses. Usually the prover wins by giving a formula for δ in terms of ε.



Limit (formal, ε–δ)

$\lim_{x \to a} f(x) = L$ means: for every $\varepsilon > 0$ there exists a $\delta > 0$ such that

$$0 < |x - a| < \delta \quad \Longrightarrow \quad |f(x) - L| < \varepsilon.$$

Here $|x - a|$ is the distance between x and a (Volume 1, Chapter 5); "$0 < |x - a|$" excludes $x = a$ itself; and $\Longrightarrow$ reads "implies". In the language of logic (Volume 3, Chapter 21): $\forall \varepsilon > 0 \; \exists \delta > 0 \; \forall x \; (0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon)$.

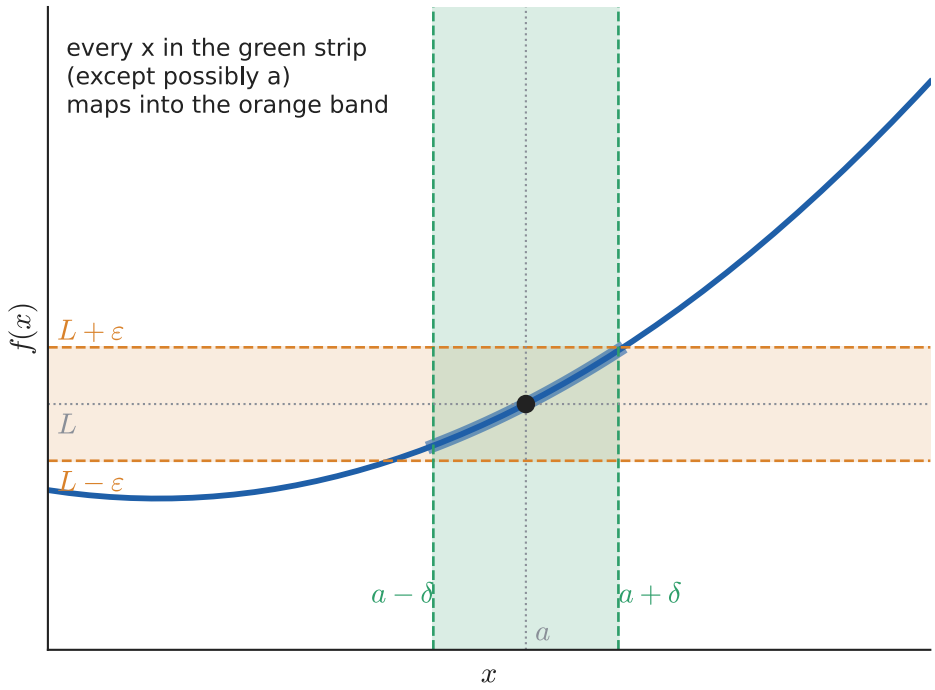


Figure 11. The ε–δ game: for any band of half-height ε around L, find a strip of half-width δ around a whose graph stays inside the band.

16.10.3. Worked examples

A linear function. Prove $\lim_{x \to 3}(2x + 1) = 7$. *Try it:* Play ε–δ for 2x + 1 .

Scratch work (finding δ): we need $|(2x + 1) - 7| < \varepsilon$. Simplify: $|2x - 6| = 2|x - 3| < \varepsilon$ exactly when $|x - 3| < \frac{\varepsilon}{2}$. So choose $\delta = \frac{\varepsilon}{2}$.

Proof: given $\varepsilon > 0$, let $\delta = \frac{\varepsilon}{2}$. If $0 < |x - 3| < \delta$, then

$$|(2x + 1) - 7| = 2|x - 3| < 2\delta = \varepsilon. \qquad \checkmark$$

The slope 2 means outputs move twice as fast as inputs, so the input window must be half as wide as the output window. If the challenger demands $\varepsilon = 0.001$, the prover answers $\delta = 0.0005$.

A quadratic. Prove $\lim_{x \to 2} x^2 = 4$. *Try it:* Play ε–δ for x² .

Scratch work: $|x^2 - 4| = |x - 2|\,|x + 2|$. The factor $|x - 2|$ is what we control; the factor $|x + 2|$ varies, so we bound it. If we insist $\delta \le 1$, then $|x - 2| < 1$ means $1 < x < 3$, so $|x + 2| < 5$. Then $|x^2 - 4| < 5|x - 2|$, which is below ε when $|x - 2| < \frac{\varepsilon}{5}$.

Proof: given $\varepsilon > 0$, let $\delta = \min\left(1, \frac{\varepsilon}{5}\right)$. If $0 < |x - 2| < \delta$, then $|x + 2| < 5$ (because $\delta \le 1$) and

$$|x^2 - 4| = |x - 2|\,|x + 2| < \frac{\varepsilon}{5} \cdot 5 = \varepsilon. \qquad \checkmark$$

Sequences: the ε–N version. For sequences the "closeness of input" is replaced by "far enough along": $\lim_{n \to \infty} a_n = L$ means for every $\varepsilon > 0$ there is an N such that $|a_n - L| < \varepsilon$ for all $n > N$. Example: $\frac{1}{n} \to 0$, because given ε, any $N \ge \frac{1}{\varepsilon}$ works: $n > N$ gives $\frac{1}{n} < \frac{1}{N} \le \varepsilon$.

Computer-science connection: ε–δ is a specification

The ε–δ definition reads like an API contract or a property-based test: "for every tolerance the client requests, the implementation can supply an input precision that guarantees it". The quantifier order matters exactly as in code: δ may depend on ε (the server sees the request before responding), but must work for *all* x in the window (no cherry-picking test cases). Swapping "for every ε there exists δ" to "there exists δ for every ε" changes the meaning entirely — the same kind of bug as confusing `all(any(…))` with `any(all(…))` (Volume 3, Chapter 21). Numerical analysts use precisely this style of reasoning to state error guarantees for algorithms (Chapter 32).



The ε–δ game

The app plays challenger: it shows a graph, a point a, a claimed limit L, and a horizontal band of half-height ε. The user drags a vertical strip of half-width δ; the portion of the graph inside the strip turns green if it fits within the band and red where it escapes. Each round, ε shrinks. For a function with a jump, the user discovers that no δ works once ε is smaller than half the jump — a visual proof that the limit does not exist.

Try it: Try to win at a jump · Play the ε–δ game

16.11. Limits meet the computer

Limits are about what happens as something gets *arbitrarily* small or large. Computers have neither arbitrarily small nor arbitrarily large numbers. This gap is where many numerical bugs live.

16.11.1. Why you cannot just plug in h = 1e-20

The car example suggests a tempting shortcut for computing a derivative: just evaluate the average rate of change with a tiny h. Try it on $\sin$ at $x = 1$, where the true answer is $\cos 1 \approx 0.5403023$: *Try it:* Shrink h yourself .

PYTHON

```
import math

x = 1.0
exact = math.cos(x)              # the true rate of change of sin at x = 1
for k in [1, 2, 4, 6, 8, 10, 12, 14, 16, 20]:
    h = 10.0 ** (-k)
    approx = (math.sin(x + h) - math.sin(x)) / h
    print(f"h = 1e-{k:<2}  approx = {approx:.15f}  error = {abs(approx - exact):.1e}")
print(1.0 + 1e-20 == 1.0)
```

```
h = 1e-1   approx = 0.497363752535389  error = 4.3e-02
h = 1e-2   approx = 0.536085981011869  error = 4.2e-03
h = 1e-4   approx = 0.540260231418621  error = 4.2e-05
h = 1e-6   approx = 0.540301885121330  error = 4.2e-07
h = 1e-8   approx = 0.540302291796024  error = 1.4e-08
h = 1e-10  approx = 0.540302247387103  error = 5.8e-08
h = 1e-12  approx = 0.540345546085064  error = 4.3e-05
h = 1e-14  approx = 0.544009282066327  error = 3.7e-03
h = 1e-16  approx = 0.000000000000000  error = 5.4e-01
h = 1e-20  approx = 0.000000000000000  error = 5.4e-01
True
```

Mathematically, shrinking h always helps. Computationally, the error falls until $h \approx 10^{-8}$ and then *rises*, until at $h = 10^{-16}$ the answer is 0 — completely wrong. Two errors compete:

- **Truncation error** (the mathematics): the secant slope differs from the tangent slope by an amount roughly proportional to h. Shrinking h reduces it.
- **Rounding error** (the computer): a double carries about 16 significant digits (machine epsilon $\approx 2.2 \times 10^{-16}$). $\sin(1+h)$ and $\sin 1$ agree in their first many digits, so subtracting them cancels those digits and leaves mostly noise (**catastrophic cancellation**); dividing by a tiny h then magnifies that noise, roughly like $\frac{10^{-16}}{h}$.

At $h = 10^{-20}$ it is worse still: $1 + 10^{-20}$ rounds to exactly 1.0 (the last line prints `True`), so the numerator is exactly zero. The best compromise is around $h \approx \sqrt{\text{machine epsilon}} \approx 10^{-8}$, giving only about 8 correct digits. Better formulas (central differences), and better approaches altogether (automatic differentiation, as used in every ML framework), are the subject of Chapter 32.

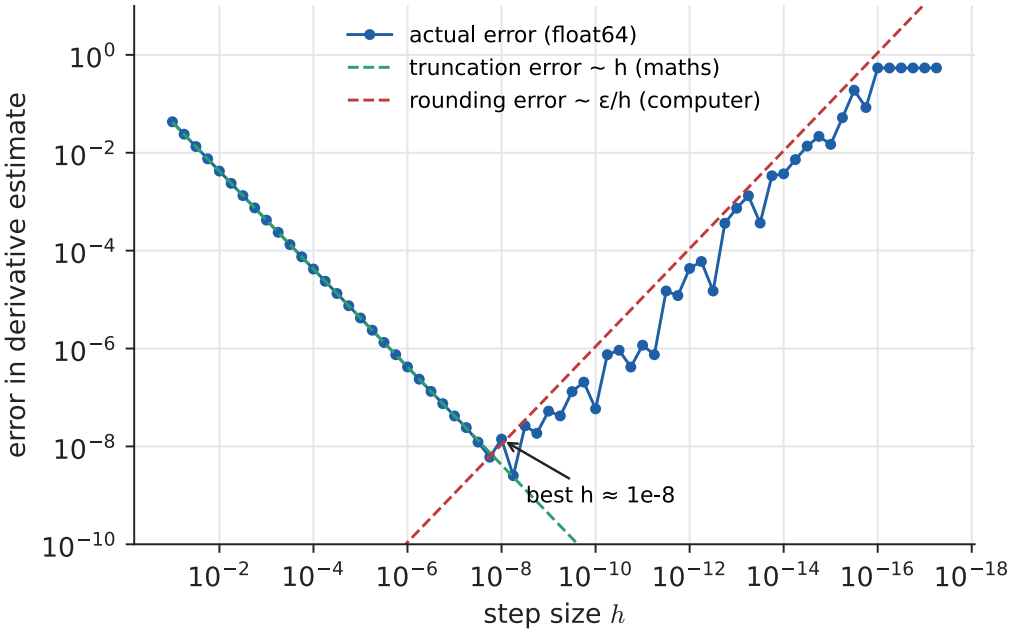


Figure 12. Error in a computed derivative versus step size: the mathematical error shrinks with h, the rounding error grows, and the best you can do is in the middle.

The $(1 + \frac{1}{n})^n \to e$ limit fails in the same way: *Try it:* Watch (1 + 1/n)ⁿ collapse .

```python
import math
for k in [1, 3, 6, 9, 12, 15, 16, 17]:
    n = 10**k
    print(f"n = 1e{k:<2}  (1 + 1/n)^n = {(1 + 1/n)**n:.12f}")
print("e          =", f"{math.e:.12f}")
```

```
n = 1e1   (1 + 1/n)^n = 2.593742460100
n = 1e3   (1 + 1/n)^n = 2.716923932236
n = 1e6   (1 + 1/n)^n = 2.718280469096
n = 1e9   (1 + 1/n)^n = 2.718282052012
n = 1e12  (1 + 1/n)^n = 2.718523496037
n = 1e15  (1 + 1/n)^n = 3.035035206549
n = 1e16  (1 + 1/n)^n = 1.000000000000
n = 1e17  (1 + 1/n)^n = 1.000000000000
e          = 2.718281828459
```

Approaching e nicely, then drifting away, then collapsing to 1 once $1 + 10^{-16}$ rounds to 1. **The mathematical limit is a statement about exact arithmetic. Floating point approximates it well only in a middle range.** This is why library functions like `math.log1p` and `math.expm1` exist: they compute $\ln(1 + x)$ and $e^x - 1$ accurately for tiny x without forming $1 + x$.



Precision runs out before the limit does

Whenever code takes a limit by brute force — a tiny h, a huge n, iterating "until it converges" — check for cancellation (subtracting nearly equal numbers) and absorption (adding a tiny number to a large one). Plotting the error against the parameter on log–log axes, as above, is the quickest diagnosis: a V shape means rounding error has taken over.

16.11.2. Convergence of iterative algorithms

Many algorithms produce a sequence of approximations $x_0, x_1, x_2, \ldots$ that should converge to the answer: square roots, Newton's method, gradient descent, PageRank, iterative linear solvers. "The algorithm converges" means precisely $\lim_{n \to \infty} x_n = x^*$. Heron's ancient method for $\sqrt{a}$ iterates $x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$ (if x_n is too big, $\frac{a}{x_n}$ is too small, so average them):

```python
def heron_sqrt(a, x0=1.0, tol=1e-15, max_iter=60):
    x = x0
    for n in range(1, max_iter + 1):
        x_new = (x + a / x) / 2
        print(n, f"{x_new:.16f}", f"{abs(x_new - x):.1e}")
        if abs(x_new - x) <= tol * abs(x_new):
            return x_new
        x = x_new
    raise RuntimeError("did not converge")

heron_sqrt(2)
```

Try it: Step Heron's method for $\sqrt{2}$.

```
1 1.5000000000000000 5.0e-01
2 1.4166666666666665 8.3e-02
3 1.4142156862745097 2.5e-03
4 1.4142135623746899 2.1e-06
5 1.4142135623730949 1.6e-12
6 1.4142135623730949 0.0e+00
```

Three practical lessons follow from the limit concept:

1. **Stopping criteria.** We never reach the limit; we stop when successive iterates differ by less than a tolerance, ideally a *relative* one (`tol * abs(x_new)`), since an absolute tolerance of 10^{-15} is meaningless for numbers near 10^{20}.
2. **Guard against non-convergence.** A `max_iter` cap turns "this sequence has no limit" from an infinite loop into an error.
3. **Rate of convergence.** The step sizes above go $5 \times 10^{-1}, 8 \times 10^{-2}, 2.5 \times 10^{-3}, 2 \times 10^{-6}, 1.6 \times 10^{-12}$: the number of correct digits doubles each step (**quadratic convergence**). Bisection, by contrast, gains a constant factor of 2 per step (**linear convergence**). These rates are themselves defined by limits, such as $\lim \frac{|e_{n+1}|}{|e_n|^2}$, and are studied in Chapter 32.

16.11.3. Asymptotic analysis is limits at infinity

Big-O notation, used informally in Volume 1, Chapter 14, is really a statement about limits at infinity. To compare a cost function $f(n)$ with a reference function $g(n)$, look at the ratio:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & \Rightarrow\ f \text{ grows strictly slower: } f = o(g) \text{ (and so } f = O(g)) \\ c,\ 0 < c < \infty & \Rightarrow\ f \text{ and } g \text{ grow at the same rate: } f = \Theta(g) \\ \infty & \Rightarrow\ f \text{ grows strictly faster: } f \neq O(g). \end{cases}$$

(The precise definition, $f = O(g)$ if $f(n) \le Cg(n)$ for some constant C and all large n, does not require the limit to exist, but when it does exist, this test settles the question.)

PYTHON

```python
import math

costs = {
    "3n^2 + 10n": lambda n: 3 * n**2 + 10 * n,
    "n log2 n": lambda n: n * math.log2(n),
    "n^1.5": lambda n: n ** 1.5,
}
g = lambda n: n**2
for n in [10, 1000, 10**6]:
    row = [f"{name}: {f(n) / g(n):.6f}" for name, f in costs.items()]
    print(f"n = {n:>7}  ratio to n^2 ->  " + "   ".join(row))
```

```
n =      10  ratio to n^2 ->  3n^2 + 10n: 4.000000   n log2 n: 0.332193   n^1.5: 0.316228
n =    1000  ratio to n^2 ->  3n^2 + 10n: 3.010000   n log2 n: 0.009966   n^1.5: 0.031623
n = 1000000  ratio to n^2 ->  3n^2 + 10n: 3.000010   n log2 n: 0.000020   n^1.5: 0.001000
```

$\frac{3n^2+10n}{n^2} = 3 + \frac{10}{n} \to 3$, so $3n^2 + 10n = \Theta(n^2)$: the "divide by the highest power" technique, applied to algorithm costs. $\frac{n \log n}{n^2} = \frac{\log n}{n} \to 0$, so $n \log n = o(n^2)$: the growth hierarchy (logs lose to powers) is why $O(n \log n)$ sorting beats $O(n^2)$ sorting for large inputs, whatever the constant factors.

Computer-science connection: limits throughout computing

- **Algorithm analysis:** O, Θ, o are statements about $\lim_{n\to\infty} f(n)/g(n)$. Constant factors and lower-order terms vanish in the limit — which is both the power of asymptotic analysis and its blind spot (an $O(n)$ algorithm with a constant of 10^6 loses to an $O(n^2)$ one for all $n < 10^6$).
- **Numerical computing:** derivatives, integrals, and series are limits that must be approximated with finite precision (Chapter 32).
- **Machine learning:** "training converged" means the loss sequence approached a limit; learning-rate schedules are designed so that the parameter sequence converges.
- **Continuity in software:** graphics and animation need continuous motion (no jumps between frames); control systems need continuous responses; and a discontinuity in a cost function is why some optimisers fail.
- **Bisection everywhere:** `git bisect`, binary search, and binary-search-on-the-answer problems are all the IVT's bisection in discrete form.



Floating-point limits

A slider for h from 10^{-1} to 10^{-20} (log scale) computes the difference quotient of a chosen function in double precision and in exact rational arithmetic side by side, plotting both errors on log–log axes. Show the bits of $x + h$ and x as h shrinks, highlighting the moment they become identical.

Try it: Floating-point limits, bit by bit



Who invented the limit?

Archimedes (c. 250 BCE) computed the area of a parabolic segment and bounds for π by squeezing the unknown between ever-closer inner and outer polygons — the squeeze theorem in all but name. Newton spoke of "prime and ultimate ratios" of "evanescent quantities"; Leibniz of infinitesimals. Cauchy (1821) put the limit at the centre of calculus, and Weierstrass (1850s–60s) gave the ε–δ form used today. Infinitesimals made a comeback in 1961, when Abraham Robinson's **non-standard analysis** made them rigorous — so Leibniz's intuition was vindicated after 300 years.

US curriculum

Continuity, its three-part definition, types of discontinuity, the intermediate value theorem, limits at infinity, and asymptotes are all part of **AP Calculus AB/BC** Unit 1; asymptotes of rational functions are also covered in **Algebra II** and **Precalculus**. The squeeze theorem and $\lim \frac{\sin x}{x} = 1$ are standard AP content. Bisection appears in some **Precalculus** courses and in college **numerical analysis**; asymptotic notation belongs to college **algorithms** courses.

16.12. Interactive tools

Browser tools on the companion website that go with this chapter. Each example link in the chapter opens one of them with that example already set up.

- Calculus, visually
- Graphing calculator
- Symbolic algebra
- Root finding, step by step
- Where floating point bites

16.13. Exercises

Recognition

1. Use the graph of $f(x) = \frac{|x|}{x}$ to state $\lim_{x \to 0^-} f(x)$, $\lim_{x \to 0^+} f(x)$, and whether $\lim_{x \to 0} f(x)$ exists.
2. For each, state which condition of continuity fails at the given point and name the type of discontinuity: (a) $\frac{x^2 - 1}{x - 1}$ at 1 (b) $\lfloor x \rfloor$ at 3 (c) $\frac{1}{(x-2)^2}$ at 2.
3. Which of these limits are indeterminate forms on substitution? (a) $\lim_{x \to 1} \frac{x^2 - 1}{x - 1}$ (b) $\lim_{x \to 1} \frac{x^2 + 1}{x - 1}$ (c) $\lim_{x \to 0} \frac{\tan x}{x}$.

Application

4. Evaluate: (a) $\lim_{x \to -2} \frac{x^2 + 5x + 6}{x + 2}$ (b) $\lim_{x \to 4} \frac{\sqrt{x} - 2}{x - 4}$ (c) $\lim_{h \to 0} \frac{(3+h)^2 - 9}{h}$.
5. Evaluate: (a) $\lim_{x \to 0} \frac{\sin 5x}{\sin 2x}$ (b) $\lim_{x \to 0} \frac{\tan x}{x}$ (c) $\lim_{x \to 0} \frac{1 - \cos x}{x^2}$.
6. Evaluate: (a) $\lim_{x \to \infty} \frac{4x^3 - x}{2x^3 + 5x^2}$ (b) $\lim_{x \to \infty} \frac{7x + 2}{x^2 + 1}$ (c) $\lim_{x \to -\infty} \frac{x^2}{3x + 1}$.
7. Find all vertical and horizontal asymptotes of $f(x) = \frac{x^2 - 4}{x^2 - 3x + 2}$.
8. Find c so that $f(x) = x^2$ for $x < 1$, $f(x) = 3x - c$ for $x \geq 1$ is continuous.

Problem solving

9. Show that $\cos x = x$ has a solution in $[0, \frac{\pi}{2}]$, then perform three steps of bisection by hand. *Try it:* Check your three bisection steps .
10. How many bisection steps are needed to locate a root in $[0, 4]$ to within 10^{-6}?
11. Use the squeeze theorem to find $\lim_{x \to \infty} \frac{\sin x}{x}$.
12. (ε–δ) Prove that $\lim_{x \to -1}(5 - 3x) = 8$. What δ works for $\varepsilon = 0.03$?
13. A program computes $f'(x)$ using $\frac{f(x+h)-f(x)}{h}$ with $h = 10^{-15}$ and gets nonsense. Explain why, and suggest a better h.

▼ **Answers**

1. Left limit -1, right limit 1; they differ, so the two-sided limit does not exist.
2. (a) $f(1)$ is undefined (condition 1); the limit is 2 — removable. (b) $\lfloor 3 \rfloor = 3$ is defined, but the left limit is 2 and the right limit is 3, so the limit does not exist (condition 2) — jump. (c) $f(2)$ undefined and the function blows up to $+\infty$ — infinite discontinuity.
3. (a) $\frac{0}{0}$: indeterminate. (b) $\frac{2}{0}$: not indeterminate — the limit does not exist ($-\infty$ from the left, $+\infty$ from the right). (c) $\frac{0}{0}$: indeterminate.
4. (a) $\frac{(x+2)(x+3)}{x+2} = x + 3 \to 1$. (b) Multiply by $\sqrt{x}+2$: $\frac{x-4}{(x-4)(\sqrt{x}+2)} = \frac{1}{\sqrt{x}+2} \to \frac{1}{4}$. (c) $\frac{9+6h+h^2-9}{h} = 6 + h \to 6$ — this is the slope of x^2 at $x = 3$, a derivative.
5. (a) $\frac{\sin 5x}{\sin 2x} = \frac{\sin 5x}{5x} \cdot \frac{2x}{\sin 2x} \cdot \frac{5}{2} \to 1 \cdot 1 \cdot \frac{5}{2} = \frac{5}{2}$. (b) $\frac{\tan x}{x} = \frac{\sin x}{x} \cdot \frac{1}{\cos x} \to 1 \cdot 1 = 1$. (c) $\frac{1-\cos x}{x^2} = \frac{\sin^2 x}{x^2(1+\cos x)} = \left(\frac{\sin x}{x}\right)^2 \frac{1}{1+\cos x} \to 1 \cdot \frac{1}{2} = \frac{1}{2}$.
6. (a) Equal degrees: $\frac{4}{2} = 2$. (b) Numerator degree lower: 0. (c) Numerator degree higher: $\frac{x^2}{3x+1} = \frac{x}{3+1/x}$; as $x \to -\infty$ this tends to $-\infty$ (no finite limit).
7. $\frac{(x-2)(x+2)}{(x-1)(x-2)}$. The factor $x - 2$ cancels, so $x = 2$ is a hole (where the limit is $\frac{4}{1} = 4$), not an asymptote. Vertical asymptote: $x = 1$. Equal degrees, leading coefficients 1 and 1: horizontal asymptote $y = 1$.
8. Left limit at 1: $1^2 = 1$. Right limit and value: $3 - c$. Need $3 - c = 1$, so $c = 2$.
9. Let $g(x) = \cos x - x$, continuous. $g(0) = 1 > 0$, $g(\frac{\pi}{2}) = -\frac{\pi}{2} < 0$; by the IVT there is a root. Bisection on $[0, 1.5708]$: $m = 0.7854$, $g(m) = 0.7071 - 0.7854 = -0.0783 < 0 \to [0, 0.7854]$. $m = 0.3927$, $g = 0.9239 - 0.3927 = 0.5312 > 0 \to [0.3927, 0.7854]$. $m = 0.5890$, $g = 0.8315 - 0.5890 = 0.2425 > 0 \to [0.5890, 0.7854]$. (The root is about 0.7391.)
10. Need $\frac{4}{2^n} \le 10^{-6}$, i.e. $2^n \ge 4 \times 10^6$, $n \ge \log_2(4 \times 10^6) \approx 21.9$. So 22 steps (one or two fewer if you return the midpoint and count half-widths).
11. For $x > 0$: $-\frac{1}{x} \le \frac{\sin x}{x} \le \frac{1}{x}$, and both bounds tend to 0. So the limit is 0. (Contrast $x \to 0$, where the limit is 1.)
12. $|(5 - 3x) - 8| = |-3x - 3| = 3|x + 1|$. Given ε, take $\delta = \frac{\varepsilon}{3}$: then $0 < |x + 1| < \delta$ gives $|(5 - 3x) - 8| < 3\delta = \varepsilon$. For $\varepsilon = 0.03$, $\delta = 0.01$ (or anything smaller).

13. $f(x+h)$ and $f(x)$ agree in nearly all of their roughly 16 significant digits, so their difference is dominated by rounding error of size about $10^{-16}|f(x)|$; dividing by $h = 10^{-15}$ makes that error of order 0.1 — as large as the answer. Use $h \approx 10^{-8}$ (times the scale of x), or a central difference $\frac{f(x+h)-f(x-h)}{2h}$ with $h \approx 10^{-5}$, or automatic differentiation.

16.14. Mastery check

1. Explain in two or three sentences why the instantaneous speed of a car is defined as a limit, and why we cannot simply substitute $h = 0$ into the average-speed formula.
2. Evaluate $\lim_{x \to 5} \dfrac{x^2 - 25}{x^2 - 4x - 5}$.
3. Evaluate $\lim_{x \to 0} \dfrac{\sqrt{1+x} - 1}{x}$.
4. Evaluate $\lim_{x \to 0} \dfrac{\sin 4x}{3x}$ and $\lim_{x \to \infty} \dfrac{5x^2 - 3}{2 - x^2}$.
5. State the three conditions for continuity at a, and decide whether $f(x) = \frac{x^2 - x}{x}$ (for $x \neq 0$), $f(0) = -1$ is continuous at 0.
6. Use the IVT to show $x^5 + x - 1 = 0$ has a root in $[0, 1]$, and give the first two bisection intervals.
7. Find $\lim_{n \to \infty} \dfrac{2n^2 + n \log_2 n}{n^2}$, and say what it tells you about the running time $2n^2 + n \log_2 n$.

Decision rule: All correct without hesitation → move on to Chapter 17. Stuck on 1 → reread Section 16.1 and Section 16.2. Stuck on 2–4 → reread Section 16.5, Section 16.6, and Section 16.9. Stuck on 5–6 → reread Section 16.7 and Section 16.8. Then work the section exercises.

▼ **Mastery check answers**

1. Instantaneous speed is not a ratio of two measurable changes (over an instant both are zero), so we define it as the value the average speed $\frac{d(t+h)-d(t)}{h}$ approaches as the interval h shrinks. Substituting $h = 0$ gives $\frac{0}{0}$, which is meaningless; the limit asks what happens *near* $h = 0$, never at it.
2. $\frac{(x-5)(x+5)}{(x-5)(x+1)} = \frac{x+5}{x+1} \to \frac{10}{6} = \frac{5}{3}$.
3. Multiply by the conjugate: $\frac{(1+x)-1}{x(\sqrt{1+x}+1)} = \frac{1}{\sqrt{1+x}+1} \to \frac{1}{2}$.
4. $\frac{\sin 4x}{3x} = \frac{4}{3} \cdot \frac{\sin 4x}{4x} \to \frac{4}{3}$. Dividing by x^2: $\frac{5-3/x^2}{2/x^2-1} \to \frac{5}{-1} = -5$.
5. (1) $f(a)$ defined; (2) $\lim_{x \to a} f(x)$ exists; (3) they are equal. Here $f(0) = -1$ is defined, and for $x \neq 0$, $f(x) = x - 1 \to -1$. The limit equals $f(0)$, so f is continuous at 0. (The definition of $f(0)$ repairs what would otherwise be a removable discontinuity.)

6. $g(x) = x^5 + x - 1$ is a polynomial, hence continuous; $g(0) = -1 < 0$, $g(1) = 1 > 0$, so a root exists in $(0, 1)$. $g(0.5) = 0.03125 + 0.5 - 1 = -0.469 < 0 \rightarrow [0.5, 1]$;
$g(0.75) = 0.2373 + 0.75 - 1 = -0.0127 < 0 \rightarrow [0.75, 1]$. (The root is about 0.7549.)
7. $2 + \frac{\log_2 n}{n} \rightarrow 2 + 0 = 2$. A finite, nonzero limit: the running time is $\Theta(n^2)$; the $n \log n$ term is negligible for large n.

Chapter 17. The Derivative

Chapter map: we start with a car and a stopwatch and ask what "speed at an instant" can possibly mean. Chasing that question turns average rates of change into a limit, and the limit is the derivative. Only once the idea exists do we build the machinery: computing derivatives from first principles, deriving the power, product, quotient and chain rules, finding the derivatives of sine, cosine, e^x and $\ln x$, differentiating implicitly, and taking second and higher derivatives. We finish by asking where derivatives fail to exist, and how computers actually compute them, numerically, with dual numbers, and by backpropagation, which turns out to be nothing but the chain rule.

Why this chapter matters: almost every question of the form "how fast?" or "how sensitive?" is a derivative. How fast is the coffee cooling, how much does one extra unit cost to make, how quickly is the ladder sliding, which way should each of a billion neural-network weights move to reduce the error? Speedometers, physics engines, economists' "marginal" anything and every machine-learning training loop compute derivatives all day long.

Where this chapter sits

You need: limits and the idea of "approaching without arriving" (Chapter 16); slope of a line as rise over run (Volume 1, Chapter 5); function notation and composition (Volume 1, Chapter 11); expanding $(x+h)^n$ (Volume 1, Chapter 7); exponentials, logarithms and the number e (Volume 1, Chapter 12); sine, cosine and radians (Volume 1, Chapter 13).

This chapter unlocks: optimisation, approximation and Newton's method (Chapter 18); integration and the Fundamental Theorem of Calculus (Chapter 19); partial derivatives and gradients (Chapter 20); numerical methods (Chapter 32); and the machine-learning material in Volume 3, Chapter 33 .

17.1. Instantaneous speed: the problem

17.1.1. The problem

A car is travelling along a straight road. You have its odometer reading at every moment. Between 2:00 and 2:30 it covered 30 miles. Its **average speed** over that half-hour was

$$\text{average speed} = \frac{\text{distance travelled}}{\text{time taken}} = \frac{30 \text{ miles}}{0.5 \text{ hours}} = 60 \text{ mph}.$$

That is uncontroversial. Now look at the speedometer. At precisely 2:17:04 the needle points at 63 mph. What does that *mean*?

Speed is distance divided by time. But at a single instant, no time passes and no distance is covered. The formula gives $0/0$, which is meaningless. Yet the speedometer is not lying: the car really is going 63 mph "right now". Something about that statement is true, and this chapter is about making it precise.

This was a live philosophical problem for two thousand years (Zeno's arrow paradox is a version of it). Newton and Leibniz resolved it in the 1660s–1680s, and the resolution is the derivative.



Rate of change and the symbol Δ

The Greek capital **delta**, Δ, means "change in". If a quantity goes from d_1 to d_2, then $\Delta d = d_2 - d_1$. It is a single symbol: Δt is not $\Delta \times t$.

The **average rate of change** of d with respect to t over an interval is $\dfrac{\Delta d}{\Delta t}$. For distance against time this is average speed; for any function it is the slope of the straight line joining the two endpoints of the graph.

17.1.2. Intuition: shrink the interval

We can't divide by zero time, but we can divide by *small* times. Let a ball roll down a ramp so that its distance (metres) after t seconds is

$$d(t) = 5t^2.$$

How fast is it going at exactly $t = 2$? Compute the average speed from $t = 2$ to $t = 2 + \Delta t$ for smaller and smaller Δt:

Δt	$d(2 + \Delta t)$	$\Delta d = d(2 + \Delta t) - 20$	average speed $\Delta d / \Delta t$
1	45	25	25
0.1	22.05	2.05	20.5
0.01	20.2005	0.2005	20.05
0.001	20.020005	0.020005	20.005

The averages are homing in on 20 m/s. And we can see *why* with algebra instead of a calculator. For any $\Delta t \neq 0$:

$$\begin{aligned} \frac{\Delta d}{\Delta t} &= \frac{5(2 + \Delta t)^2 - 5 \cdot 2^2}{\Delta t} \\ &= \frac{5(4 + 4\Delta t + (\Delta t)^2) - 20}{\Delta t} && \text{(expand the square)} \\ &= \frac{20\,\Delta t + 5(\Delta t)^2}{\Delta t} && \text{(the 20s cancel)} \\ &= 20 + 5\,\Delta t && \text{(divide by } \Delta t \neq 0\text{).} \end{aligned}$$

The table is just this formula evaluated at $\Delta t = 1, 0.1, 0.01, 0.001$. As $\Delta t \to 0$, the average speed approaches 20. We *define* the instantaneous speed at $t = 2$ to be that limit:

$$v(2) = \lim_{\Delta t \to 0} \frac{d(2+\Delta t) - d(2)}{\Delta t} = \lim_{\Delta t \to 0} (20 + 5\,\Delta t) = 20 \text{ m/s}.$$

Notice the trick that dissolves the paradox. We never divide by zero. We divide by small non-zero numbers, simplify while that is legal, and only *then* ask what happens as the interval shrinks. That is exactly what limits were built for in Chapter 16.

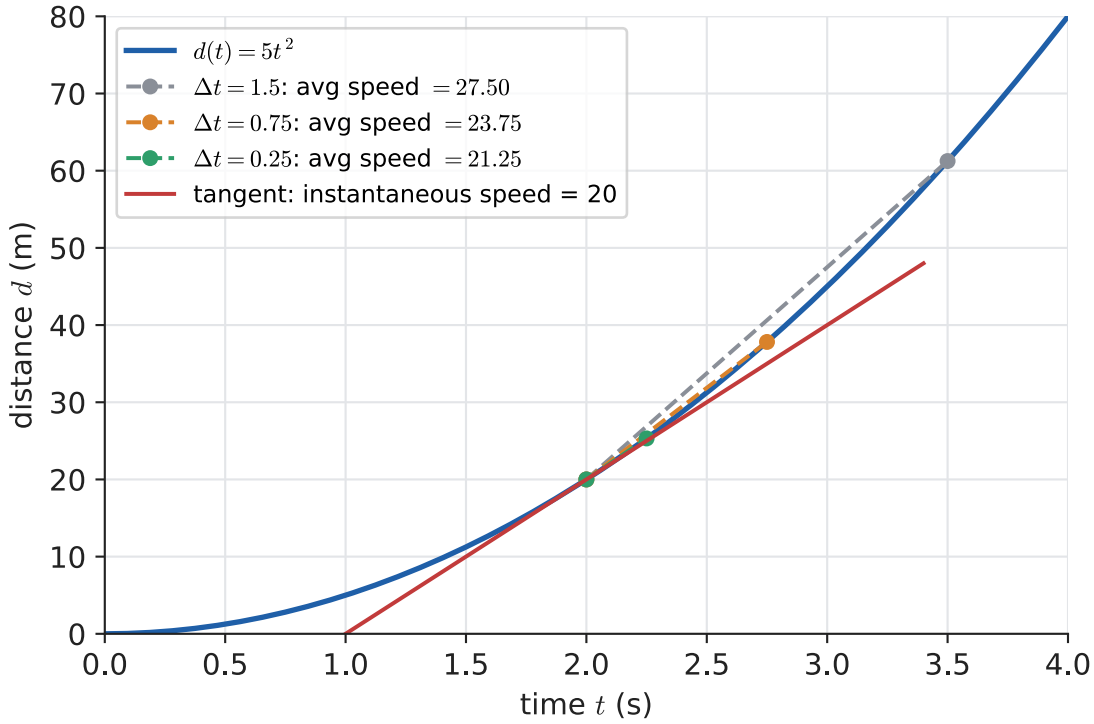


Figure 13. Average speeds over shrinking intervals approach the slope of the tangent line at $t = 2$.

17.2. From secant to tangent

17.2.1. The same idea, drawn

On a distance–time graph, the average speed between two times is the slope of the straight line joining the two points on the curve. The instantaneous speed is the slope of the line that just *touches* the curve at one point. This picture works for any function, not just motion, so let us name the pieces.

Secant line and tangent line

A **secant line** (from Latin *secare*, to cut) is a line through two points on a curve. Its slope, $\dfrac{f(a+h) - f(a)}{h}$, is the average rate of change of f between a and $a + h$.

A **tangent line** (from Latin *tangere*, to touch) at a point P is the line through P whose direction matches the curve's direction at P. Zoom in far enough on a smooth curve at P and the curve and its tangent become indistinguishable.

Note: the tangent *line* is unrelated to the trig function **tan** except by shared etymology.

Take $f(x) = \tfrac{1}{4}x^3 - x + 2$ and the point $P = (1, f(1)) = (1, 1.25)$. The secant from P to the point at $x = 1 + h$ has slope *Try it:* Drag the secant to P.

$$\begin{aligned}
\frac{f(1+h) - f(1)}{h} &= \frac{\tfrac{1}{4}(1+h)^3 - (1+h) + 2 - 1.25}{h} \\
&= \frac{\tfrac{1}{4}(1 + 3h + 3h^2 + h^3) - 1 - h + 0.75}{h} && \text{(expand } (1+h)^3\text{)} \\
&= \frac{0.25 + 0.75h + 0.75h^2 + 0.25h^3 - 0.25 - h}{h} \\
&= \frac{-0.25h + 0.75h^2 + 0.25h^3}{h} \\
&= -0.25 + 0.75h + 0.25h^2 && (h \neq 0).
\end{aligned}$$

Evaluate for a few values of h: $h = 2$ gives 2.25; $h = 1.2$ gives 1.01; $h = 0.6$ gives 0.29; $h = 0.25$ gives -0.046875. As $h \to 0$ the slope tends to -0.25, which is the slope of the tangent at P.

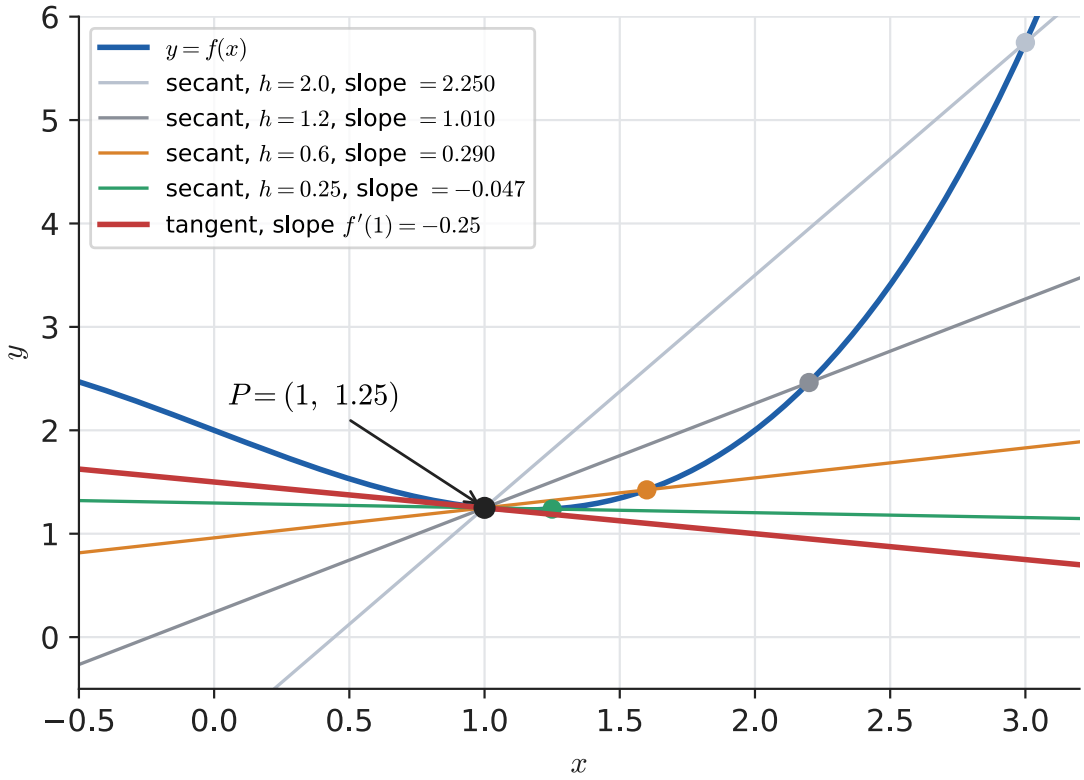


Figure 14. Secants through P swing round and settle on the tangent as the second point slides towards P.

Secant to tangent

Drag a second point Q along a curve towards a fixed point P. A readout shows the secant slope to six decimal places as h shrinks, and a second panel plots secant slope against h. Try a smooth curve, then $|x|$ at $P = (0, 0)$, and watch the slopes from the left and the right refuse to agree.

Try it: Try |x| at the origin · Open the secant-to-tangent lab · Secants of |x| at 0

17.3. The definition of the derivative

17.3.1. Formalisation

Everything so far collapses into one definition.

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

Read it piece by piece:

- $f(a+h) - f(a)$ is the change in output, Δy, when the input moves from a to $a+h$;
- dividing by h (which is Δx) gives the average rate of change, the secant slope;
- $\lim_{h \to 0}$ asks what that slope approaches as the interval shrinks, without ever setting $h = 0$;
- $f'(a)$, read "f prime of a", is the result: the **instantaneous rate of change** of f at a, the slope of the tangent there.

If the limit exists, f is **differentiable at** a. If we let a vary, we get a new function, $x \mapsto f'(x)$, the **derivative of** f. Finding it is called **differentiation**.



Derivative, difference quotient, differentiable

The **difference quotient** is $\dfrac{f(x+h) - f(x)}{h}$, the slope of a secant.

The **derivative** $f'(x)$ is the limit of the difference quotient as $h \to 0$: the slope of the tangent, or the instantaneous rate of change of f.

A function is **differentiable at** x if that limit exists (as a finite number). It is **differentiable** (on an interval) if it is differentiable at every point there.

Differentiation is the process of finding a derivative. The word comes from Leibniz's "differences", the tiny changes dx and dy.

17.3.2. Notation: four ways to write the same thing

Calculus was invented twice, by Newton and by Leibniz, and both notations survive. You will meet all of these, sometimes in the same paper.



Notations for the derivative

Lagrange (prime) notation: $f'(x)$, $f''(x)$ for the second derivative. Compact; good when the function has a name.

Leibniz notation: if $y = f(x)$, write $\frac{dy}{dx}$ ("dee y by dee x"). It looks like $\frac{\Delta y}{\Delta x}$ with the deltas made infinitely small, which is exactly the intuition. Value at a point: $\left.\frac{dy}{dx}\right|_{x=a}$.

Operator notation: $\frac{d}{dx}$ is an *operator*: a machine that takes a function and returns its derivative. $\frac{d}{dx}\left(x^2\right) = 2x$ reads "the derivative with respect to x of x^2 is $2x$". Also written D_x or D.

Newton (dot) notation: $\dot{x}$ for the derivative of x with respect to *time*, $\ddot{x}$ for the second. Common in mechanics and physics.

Leibniz notation earns its keep because it records *which variable* you are differentiating with respect to. $\frac{dV}{dr}$ and $\frac{dV}{dt}$ are different questions about the same volume V: how fast it changes per unit of radius, versus per second. It also makes the chain rule look like fraction cancellation, which (as we'll see) is a helpful mnemonic and a dangerous proof.



dy/dx is not (quite) a fraction

$\frac{dy}{dx}$ is a single symbol meaning a limit of fractions. It is not dy divided by dx, and you cannot cancel the ds to get $\frac{y}{x}$. Leibniz's notation is designed so that *many* fraction-like manipulations give correct answers (the chain rule, inverse functions, separation of variables), but each of those is a theorem, not an algebra rule. Similarly $\frac{d}{dx}$ on its own is an instruction, like a function name waiting for an argument.

17.3.3. Where you'd use this

Whenever a quantity changes, its derivative is "how fast, right now", and the difference quotient is "how fast, on average". Both show up constantly.

- **Everyday: average versus instantaneous speed.** UK average-speed cameras photograph your number plate at two points and divide. Two cameras 2 km apart and a crossing time of 72 s give $2 \div (72/3600) = 100$ km/h, about 62 mph: a secant slope. A fixed camera with radar measures (very nearly) the derivative. You can drive at 80 mph past the first camera and still average 62; the two measurements answer different questions.
- **Economics: marginal cost.** If making q units costs $C(q) = 500 + 4q + 0.01q^2$ pounds, the **marginal cost** is $C'(q) = 4 + 0.02q$. At $q = 100$ it is $C'(100) = 6$ pounds: the cost of one more unit. Check: $C(101) - C(100) = 6.01$. Economists say "marginal" where mathematicians say "derivative".

- **Science and engineering.** Electric current is the rate of flow of charge, $I = \frac{dQ}{dt}$ (one ampere is one coulomb per second). Reaction rate in chemistry is $\frac{d[\text{concentration}]}{dt}$. Power is the rate of doing work, $P = \frac{dW}{dt}$: a kettle that delivers 3 kW turns 3000 joules per second into heat.
- **Computing: rates from counters.** Monitoring systems store ever-increasing counters (total requests served) and compute rates from them. Prometheus's `rate(requests_total[5m])` is a secant slope: if the counter went from 12,000 to 15,600 over 300 seconds, the rate is $3600/300 = 12$ requests per second. Choosing the window is choosing h: a short window is closer to "instantaneous" but noisier, the same trade-off you will meet in Section 17.10.

17.4. Derivatives from first principles

Computing a derivative directly from the limit definition is called working **from first principles**. You will rarely do it once you have the rules, but doing it a few times is how you come to trust the rules, and it is how we will derive them.

Example 5. Procedure — derivative from first principles

Input: a function f.

Steps:

1. Write the difference quotient $\dfrac{f(x+h) - f(x)}{h}$.
2. Substitute the formula for f at both $x + h$ and x.
3. Simplify algebraically (expand, combine fractions, or multiply by a conjugate) until a factor of h in the numerator cancels the h in the denominator. This step is legal because $h \neq 0$ inside the limit.
4. Now that nothing divides by h, let $h \to 0$ (usually by substituting $h = 0$).

Output: $f'(x)$.

17.4.1. Worked example: $f(x) = x^2$

$$\begin{aligned} f'(x) &= \lim_{h \to 0} \frac{(x+h)^2 - x^2}{h} \\ &= \lim_{h \to 0} \frac{x^2 + 2xh + h^2 - x^2}{h} && \text{(expand)} \\ &= \lim_{h \to 0} \frac{2xh + h^2}{h} && (x^2 \text{ cancels}) \\ &= \lim_{h \to 0} (2x + h) && (\text{divide by } h \neq 0) \\ &= 2x. \end{aligned}$$

So the slope of the parabola $y = x^2$ at x is $2x$: zero at the vertex, 2 at $x = 1$, -6 at $x = -3$.

17.4.2. Worked example: $f(x) = x^3$

$$\begin{aligned} f'(x) &= \lim_{h \to 0} \frac{(x+h)^3 - x^3}{h} \\ &= \lim_{h \to 0} \frac{x^3 + 3x^2h + 3xh^2 + h^3 - x^3}{h} && \text{(binomial expansion)} \\ &= \lim_{h \to 0} \frac{3x^2h + 3xh^2 + h^3}{h} \\ &= \lim_{h \to 0} \left(3x^2 + 3xh + h^2\right) \\ &= 3x^2. \end{aligned}$$

17.4.3. Worked example: $f(x) = \dfrac{1}{x}$

Here the simplification step is "combine the fractions":

$$\begin{aligned} f'(x) &= \lim_{h \to 0} \frac{1}{h}\left(\frac{1}{x+h} - \frac{1}{x}\right) \\ &= \lim_{h \to 0} \frac{1}{h} \cdot \frac{x - (x+h)}{x(x+h)} && \text{(common denominator)} \\ &= \lim_{h \to 0} \frac{1}{h} \cdot \frac{-h}{x(x+h)} \\ &= \lim_{h \to 0} \frac{-1}{x(x+h)} && (h \text{ cancels}) \\ &= -\frac{1}{x^2} && (x \neq 0). \end{aligned}$$

The slope is always negative (the hyperbola always falls) and gets steeper near $x = 0$.

17.4.4. Worked example: $f(x) = \sqrt{x}$

Here the simplification step is the conjugate trick from Chapter 16: multiply top and bottom by $\sqrt{x+h} + \sqrt{x}$ so that the difference of square roots becomes a difference of squares.

$$\begin{aligned} f'(x) &= \lim_{h \to 0} \frac{\sqrt{x+h} - \sqrt{x}}{h} \cdot \frac{\sqrt{x+h} + \sqrt{x}}{\sqrt{x+h} + \sqrt{x}} \\ &= \lim_{h \to 0} \frac{(x+h) - x}{h\left(\sqrt{x+h} + \sqrt{x}\right)} && ((p-q)(p+q) = p^2 - q^2) \\ &= \lim_{h \to 0} \frac{1}{\sqrt{x+h} + \sqrt{x}} && (h \text{ cancels}) \\ &= \frac{1}{2\sqrt{x}} && (x > 0). \end{aligned}$$

17.4.5. Spot the pattern

Write each function as a power of x:

$f(x)$	as a power	$f'(x)$	as a power
x^2	x^2	$2x$	$2x^1$
x^3	x^3	$3x^2$	$3x^2$
$1/x$	x^{-1}	$-1/x^2$	$-1 \cdot x^{-2}$
$\sqrt{x}$	$x^{1/2}$	$1/(2\sqrt{x})$	$\frac{1}{2}x^{-1/2}$

Every time: bring the exponent down in front, then reduce the exponent by one. That is the power rule, $\frac{d}{dx}x^n = nx^{n-1}$, and we have now *earned* it rather than been handed it. The next section proves it.



Who got there first?

Newton developed his "method of fluxions" around 1665–66, while Cambridge was closed for the plague, but published little of it for decades. Leibniz developed his calculus independently in the 1670s and published first, in 1684. The resulting priority dispute became a national feud; British mathematicians clung to Newton's dot notation for a century and fell behind the Continent, which used Leibniz's far more flexible dy/dx. Notation matters.

17.4.6. Exercises

1. Use the definition to find the slope of the secant to $f(x) = x^2$ between $x = 3$ and $x = 3.1$, then find $f'(3)$.
2. Differentiate $f(x) = 4x^2 - 3x + 7$ from first principles.
3. Differentiate $f(x) = \dfrac{1}{x^2}$ from first principles and check your answer against the power-rule pattern.

▼ **Answers**

1. Secant slope $= \dfrac{3.1^2 - 3^2}{0.1} = \dfrac{9.61 - 9}{0.1} = 6.1$. From the worked example, $f'(x) = 2x$, so $f'(3) = 6$. (Formula check: secant slope $= 2x + h = 6 + 0.1$.)
2. $f(x+h) - f(x) = 4(x^2 + 2xh + h^2) - 3x - 3h + 7 - 4x^2 + 3x - 7 = 8xh + 4h^2 - 3h$. Divide by h: $8x + 4h - 3$. Let $h \to 0$: $f'(x) = 8x - 3$.
3. $\dfrac{1}{h}\left(\dfrac{1}{(x+h)^2} - \dfrac{1}{x^2}\right) = \dfrac{1}{h} \cdot \dfrac{x^2 - (x+h)^2}{x^2(x+h)^2} = \dfrac{1}{h} \cdot \dfrac{-2xh - h^2}{x^2(x+h)^2} = \dfrac{-2x - h}{x^2(x+h)^2}$. As $h \to 0$ this tends to $\dfrac{-2x}{x^4} = -\dfrac{2}{x^3}$, which is $-2x^{-3}$, exactly nx^{n-1} with $n = -2$.

17.5. The rules of differentiation

17.5.1. The problem

First principles works, but try it on $f(x) = x^7 \sin x + \sqrt{1+x^2}$ and you'll be at it all afternoon. What we want is a small set of *rules*, each derived once from the definition, that let us differentiate any formula built from standard pieces by taking it apart. This is exactly how a compiler handles expressions: a handful of rules for the building blocks and the ways of combining them, applied recursively.

The building blocks are powers, exponentials, logs and trig functions. The ways of combining are: multiply by a constant, add, multiply, divide, and compose. We need one rule for each.

17.5.2. Constant, constant multiple and sum rules

Constants. If $f(x) = c$ then $f(x+h) - f(x) = c - c = 0$, so $f'(x) = 0$. A flat line has slope zero.

Constant multiple. If $g(x) = c\,f(x)$:

$$g'(x) = \lim_{h\to 0} \frac{c\,f(x+h) - c\,f(x)}{h} = c \lim_{h\to 0} \frac{f(x+h)-f(x)}{h} = c\,f'(x).$$

Stretching a graph vertically by a factor of c stretches every slope by c.

Sum. If $s(x) = f(x) + g(x)$:

$$\begin{aligned} s'(x) &= \lim_{h\to 0} \frac{f(x+h)+g(x+h)-f(x)-g(x)}{h} \\ &= \lim_{h\to 0} \left(\frac{f(x+h)-f(x)}{h} + \frac{g(x+h)-g(x)}{h} \right) && \text{(regroup)} \\ &= f'(x) + g'(x) && \text{(limit of a sum is the sum of limits).} \end{aligned}$$

Together these say differentiation is **linear**: $\frac{d}{dx}(af + bg) = af' + bg'$. That word will mean a great deal in Chapter 27.

17.5.3. The power rule

For a positive whole number n, the binomial theorem (Volume 1, Chapter 7) expands $(x+h)^n$:

$$(x+h)^n = x^n + nx^{n-1}h + \binom{n}{2}x^{n-2}h^2 + \binom{n}{3}x^{n-3}h^3 + \cdots + h^n.$$

The only facts we need: the first term is x^n, the second is $nx^{n-1}h$, and *every remaining term contains at least* h^2. So

$$\begin{aligned} \frac{(x+h)^n - x^n}{h} &= \frac{nx^{n-1}h + \binom{n}{2}x^{n-2}h^2 + \cdots + h^n}{h} && (x^n \text{ cancels}) \\ &= nx^{n-1} + \binom{n}{2}x^{n-2}h + \cdots + h^{n-1} && (\text{divide by } h) \\ &\longrightarrow nx^{n-1} \quad \text{as } h \to 0 && (\text{every later term has a factor of } h). \end{aligned}$$

$$\boxed{\frac{d}{dx}x^n = nx^{n-1}}$$

We proved it for positive integers. It is in fact true for *every* real exponent n (we already checked $n=-1$ and $n=\frac{1}{2}$ by hand); the general proof comes for free once we have $\ln x$, later in this chapter.

Worked example. Differentiate $p(x) = 4x^5 - 3x^2 + \dfrac{2}{x} - 7\sqrt{x} + 9$. *Try it:* Differentiate it symbolically.

First rewrite every term as a power of x, because the power rule needs that form:

$$\begin{aligned} p(x) &= 4x^5 - 3x^2 + 2x^{-1} - 7x^{1/2} + 9 \\ p'(x) &= 4\cdot 5x^4 - 3\cdot 2x^1 + 2\cdot(-1)x^{-2} - 7\cdot\tfrac{1}{2}x^{-1/2} + 0 \qquad \text{(power rule, termwise)} \\ &= 20x^4 - 6x - \frac{2}{x^2} - \frac{7}{2\sqrt{x}}. \end{aligned}$$

17.5.4. The product rule

The problem. Is the derivative of a product the product of the derivatives? Try $x \cdot x = x^2$: the product of derivatives is $1\cdot 1 = 1$, but the true derivative is $2x$. So no. *Try it:* Grow the product-rule rectangle.

Intuition: a growing rectangle. Picture $f(x)\,g(x)$ as the area of a rectangle with sides f and g. Nudge x by h. The width grows by Δf and the height by Δg. The area gains three pieces: a strip $g\,\Delta f$ down the side, a strip $f\,\Delta g$ across the top, and a tiny corner $\Delta f\,\Delta g$.

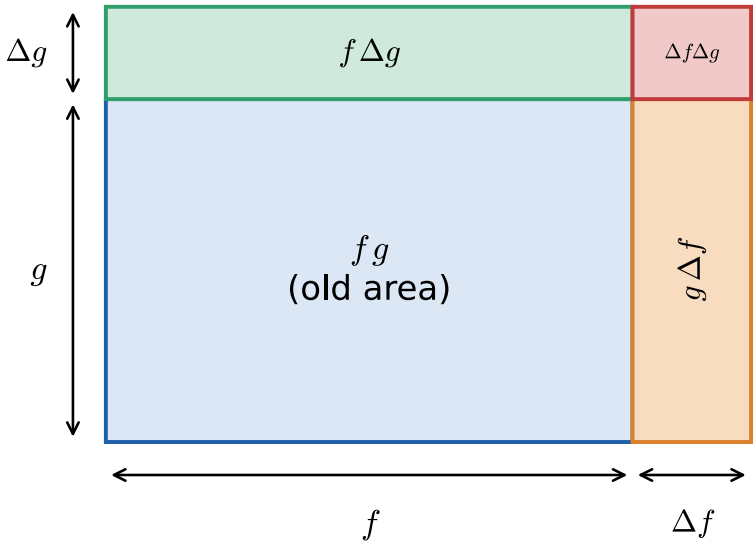


Figure 15. The product rule as a growing rectangle: the corner piece is "small times small" and vanishes in the limit.

Why it works. Divide the change in area by h:

$$\frac{\Delta(fg)}{h} = g\,\frac{\Delta f}{h} + f\,\frac{\Delta g}{h} + \frac{\Delta f}{h}\,\Delta g.$$

As $h \to 0$, $\frac{\Delta f}{h} \to f'$ and $\frac{\Delta g}{h} \to g'$, while $\Delta g \to 0$ (because g is continuous), so the corner term vanishes:

$$\boxed{(fg)' = f'g + fg'}$$

In words: "derivative of the first times the second, plus the first times derivative of the second". Each factor takes its turn changing while the other is held fixed.

Worked example. $\dfrac{d}{dx}\left(x^2 \sin x\right)$. Let $f = x^2$, $g = \sin x$, so $f' = 2x$ and $g' = \cos x$ (proved in the next section): *Try it:* Differentiate it symbolically .

$$\frac{d}{dx}\left(x^2 \sin x\right) = \underbrace{2x}_{f'} \cdot \underbrace{\sin x}_{g} + \underbrace{x^2}_{f} \cdot \underbrace{\cos x}_{g'} = 2x \sin x + x^2 \cos x.$$

17.5.5. The quotient rule

Why it works. Let $Q = f/g$, so $f = Qg$. Differentiate both sides with the product rule and solve for Q':

$$\begin{aligned} f' &= Q'g + Qg' && \text{(product rule)} \\ Q'g &= f' - Qg' = f' - \frac{f}{g}g' && \text{(substitute } Q = f/g\text{)} \\ Q' &= \frac{f' - \frac{f}{g}g'}{g} = \frac{f'g - fg'}{g^2} && \text{(divide by } g\text{, multiply top and bottom by } g\text{).} \end{aligned}$$

$$\boxed{\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}}$$

(This derivation assumes Q is differentiable; a first-principles proof removes that assumption and gives the same formula.) The traditional mnemonic: "low d-high minus high d-low, over the square of what's below". The order matters because of the minus sign.

Worked example. $\dfrac{d}{dx}\,\dfrac{x}{x^2+1}$. With $f = x$, $f' = 1$, $g = x^2 + 1$, $g' = 2x$: *Try it:* Differentiate it symbolically .

$$\begin{aligned} \frac{d}{dx}\,\frac{x}{x^2+1} &= \frac{1 \cdot (x^2+1) - x \cdot 2x}{(x^2+1)^2} \\ &= \frac{x^2 + 1 - 2x^2}{(x^2+1)^2} = \frac{1 - x^2}{(x^2+1)^2}. \end{aligned}$$

The derivative is zero at $x = \pm 1$, which is where the graph turns: the maximum value $\frac{1}{2}$ and minimum $-\frac{1}{2}$. Finding turning points this way is the subject of Chapter 18.

17.5.6. The chain rule

The problem. How do we differentiate a *composition*, a function of a function, such as $\sin(x^2)$ or $(3x^2+1)^5$? Expanding $(3x^2+1)^5$ is possible but miserable; expanding $\sqrt{1+x^2}$ is impossible.

Intuition: gears. Three gears are meshed. Gear B turns 3 times for every turn of gear A; gear C turns 2 times for every turn of gear B. How many times does C turn per turn of A? $3 \times 2 = 6$. Rates of change through a chain *multiply*.

Now write the composition as a chain. If $y = (3x^2+1)^5$, introduce the inner quantity $u = 3x^2+1$, so $y = u^5$. Then x drives u at rate $\frac{du}{dx} = 6x$, and u drives y at rate $\frac{dy}{du} = 5u^4$. So x drives y at rate

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}.$$

In Leibniz notation the dus appear to cancel, which makes the rule easy to remember. In Lagrange notation, for $y = f(g(x))$:

$$\boxed{\frac{d}{dx} f(g(x)) = f'(g(x)) \cdot g'(x)}$$

"Derivative of the outer function, evaluated at the inner function, times the derivative of the inner function."

Why it works. Write $u = g(x)$ and $\Delta u = g(x+h) - g(x)$. Provided $\Delta u \neq 0$ for small h:

$$\frac{f(g(x+h)) - f(g(x))}{h} = \frac{f(u+\Delta u) - f(u)}{\Delta u} \cdot \frac{\Delta u}{h}.$$

As $h \to 0$, $\Delta u \to 0$ (since g is continuous), so the first fraction tends to $f'(u)$ and the second to $g'(x)$. The proviso "$\Delta u \neq 0$" is a genuine gap: for some functions Δu is zero infinitely often near x. The rigorous proof patches it by replacing the first fraction with a function that equals $f'(u)$ when $\Delta u = 0$; the result is unchanged. This is the classic case where the intuitive argument is right in spirit and the careful version is a technicality.

Example 6. Procedure — the chain rule (outside-in)

Input: a composite expression $y = f(g(x))$.

Steps:

1. Identify the *outer* function f (the last operation you would perform if evaluating on a calculator) and the *inner* function g (everything fed into it).
2. Differentiate the outer function, leaving its inside untouched: $f'(g(x))$.
3. Multiply by the derivative of the inside: $g'(x)$.
4. If the inside is itself a composition, repeat steps 1–3 on it (each layer contributes one factor).

Output: $\dfrac{dy}{dx} = f'(g(x))\,g'(x)$.

Worked examples. *Try it:* Differentiate $(3x^2 + 1)^5$.

$$\frac{d}{dx}(3x^2+1)^5 = \underbrace{5(3x^2+1)^4}_{\text{outer: } u^5 \to 5u^4} \cdot \underbrace{6x}_{\text{inner}} = 30x(3x^2+1)^4,$$

$$\frac{d}{dx}\sqrt{1+x^2} = \frac{d}{dx}(1+x^2)^{1/2} = \tfrac{1}{2}(1+x^2)^{-1/2}\cdot 2x = \frac{x}{\sqrt{1+x^2}},$$

$$\frac{d}{dx}\sin(x^2) = \cos(x^2)\cdot 2x,$$

$$\frac{d}{dx}\sin^3(2x) = 3\sin^2(2x)\cdot\cos(2x)\cdot 2 = 6\sin^2(2x)\cos(2x) \qquad \text{(three layers: cube, sine, } 2x\text{)}.$$



Forgetting the inner derivative

The single most common calculus error is writing $\frac{d}{dx}\sin(x^2) = \cos(x^2)$ and stopping. Every layer of a composition contributes a factor. A quick sanity check: if you replace x by $2x$, the graph is squeezed horizontally by a factor of 2, so every slope must double. The chain rule's "times the derivative of the inside" is exactly that factor.

Also: $(fg)' \neq f'g'$ and $(f/g)' \neq f'/g'$. Products and quotients have their own rules.

17.5.7. Where you'd use this

The chain rule is the one you use without noticing, because it is how **rates convert between units**.

- **Everyday.** Your car uses 0.06 litres per kilometre and you are driving at 90 km/h. Fuel per hour is $\frac{dF}{dt} = \frac{dF}{dx}\cdot\frac{dx}{dt} = 0.06 \times 90 = 5.4$ litres per hour. Every "per" conversion (pence per kWh times kWh per hour, heartbeats per minute times minutes per day) is the chain rule with constant rates;

calculus says it still works when the rates vary.

- **Science and engineering: related rates.** The balloon exercise below is the template: you know how fast one quantity changes (radius) and want another (volume) linked to it by a formula. Pumping air into a balloon, oil spreading in a circular slick ($A = \pi r^2$, so $\frac{dA}{dt} = 2\pi r \frac{dr}{dt}$), the shadow of a walker under a streetlight: all one chain rule.
- **Computing: animation easing.** Animators move objects with "ease-in, ease-out" curves rather than constant speed. A standard one is **smoothstep**, $s(t) = 3t^2 - 2t^3$ for $0 \le t \le 1$. Its derivative $s'(t) = 6t - 6t^2$ is 0 at both ends (the object starts and stops gently) and peaks at $s'(0.5) = 1.5$ in the middle. To run a 2-second animation you use $s(t/2)$, and the chain rule tells you its speed is $\frac{1}{2}s'(t/2)$. The product rule, meanwhile, is behind every "growth of a product" question: revenue = price × quantity, so $R' = p'q + pq'$, which is why a price rise that cuts sales can still lose money.

17.5.8. Choosing a method

You will often have a choice of route. Some guidance:

- **Simplify first.** $\dfrac{x^2 + 1}{x}$ can be done with the quotient rule, but rewriting as $x + x^{-1}$ gives $1 - x^{-2}$ in one line.
- **Constant denominators are not quotients.** $\dfrac{x^3}{6}$ is $\frac{1}{6}x^3$; derivative $\frac{1}{2}x^2$. Using the quotient rule works but invites errors.
- **Roots and reciprocals become powers.** $\dfrac{1}{\sqrt{x}} = x^{-1/2}$, derivative $-\frac{1}{2}x^{-3/2}$.
- **Quotient rule versus product-plus-chain.** $\dfrac{f}{g} = f \cdot g^{-1}$; some people never use the quotient rule at all and instead apply the product rule and chain rule to $f \cdot g^{-1}$. Both are correct; the quotient rule's formula is usually tidier.
- **Logarithmic differentiation** (take $\ln$ of both sides, then differentiate implicitly) is best for messy products of powers like x^x or $\dfrac{(x+1)^3\sqrt{x-2}}{(x^2+5)^4}$. See the exercises below.

Rules of differentiation

Rule	Formula
Constant	$\frac{d}{dx}c = 0$
Power	$\frac{d}{dx}x^n = nx^{n-1}$ (any real n)
Constant multiple	$(cf)' = cf'$
Sum / difference	$(f \pm g)' = f' \pm g'$
Product	$(fg)' = f'g + fg'$
Quotient	$(f/g)' = (f'g - fg')/g^2$
Chain	$\frac{d}{dx}f(g(x)) = f'(g(x))\,g'(x)$, or $\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$

Differentiation is an algorithm; integration is not

The rules above are *complete* for elementary functions: any formula built from constants, x, arithmetic, powers, exp, log and trig can be differentiated mechanically by recursing on its expression tree. That is how symbolic systems such as SymPy or Mathematica do it: a pattern-match per node type, emitting a new tree. (The output can be large, so they then run a simplifier.)

The reverse operation, integration (Chapter 19), has no such complete rule set. Many innocent-looking functions, such as e^{-x^2}, have no elementary antiderivative at all. Differentiation takes things apart; integration has to guess how they were put together.

17.5.9. Exercises

1. Differentiate $y = 6x^4 - \dfrac{3}{x^3} + 5\sqrt[3]{x}$.
2. Differentiate $y = (2x - 1)(x^3 + 4)$ twice: once with the product rule and once by expanding first. Confirm they agree.
3. Differentiate $y = \dfrac{x^2}{x - 3}$.
4. Differentiate $y = (x^2 - 5x)^{10}$ and $y = \dfrac{1}{(4 - x^2)^3}$.
5. A spherical balloon's volume is $V = \frac{4}{3}\pi r^3$, and its radius grows as $r(t) = 1 + 0.5t$ cm. Use the chain rule to find $\frac{dV}{dt}$ at $t = 2$ s.

▼ **Answers**

1. Rewrite: $y = 6x^4 - 3x^{-3} + 5x^{1/3}$. Then
   $y' = 24x^3 + 9x^{-4} + \frac{5}{3}x^{-2/3} = 24x^3 + \dfrac{9}{x^4} + \dfrac{5}{3\sqrt[3]{x^2}}$.
2. Product rule: $2(x^3+4) + (2x-1)\cdot 3x^2 = 2x^3 + 8 + 6x^3 - 3x^2 = 8x^3 - 3x^2 + 8$.
   Expanding: $y = 2x^4 - x^3 + 8x - 4$, so $y' = 8x^3 - 3x^2 + 8$. They agree.
3. Quotient rule with $f = x^2, f' = 2x, g = x - 3, g' = 1$:
   $y' = \dfrac{2x(x-3) - x^2}{(x-3)^2} = \dfrac{x^2 - 6x}{(x-3)^2} = \dfrac{x(x-6)}{(x-3)^2}$.
4. $10(x^2 - 5x)^9(2x - 5)$. For the second, write $(4 - x^2)^{-3}$: derivative
   $-3(4-x^2)^{-4}\cdot(-2x) = \dfrac{6x}{(4-x^2)^4}$.
5. $\frac{dV}{dr} = 4\pi r^2$, $\frac{dr}{dt} = 0.5$. At $t = 2$, $r = 2$, so $\frac{dV}{dt} = 4\pi(2)^2(0.5) = 8\pi \approx 25.1$ cm³/s.

> **US curriculum**
>
> The limit definition, the power, product, quotient and chain rules, and the derivatives of trig, exponential and log functions are the core of **AP Calculus AB** (or a first-semester college **Calculus I**), typically grade 11 or 12. Some **Precalculus** courses preview the difference quotient. In the UK this material was A-level Pure Mathematics.

17.6. Derivatives of sine, cosine, e^x and $\ln x$

17.6.1. Sine and cosine

Intuition first. Look at the graph of $\sin x$ and read off its slope. At $x = 0$ it rises steeply (slope 1). At $x = \pi/2$, the peak, the slope is 0. At $x = \pi$ it falls as steeply as it rose (slope -1). At $3\pi/2$ it bottoms out (slope 0). Plot those slopes and you get $1, 0, -1, 0, 1, \ldots$: the cosine curve.

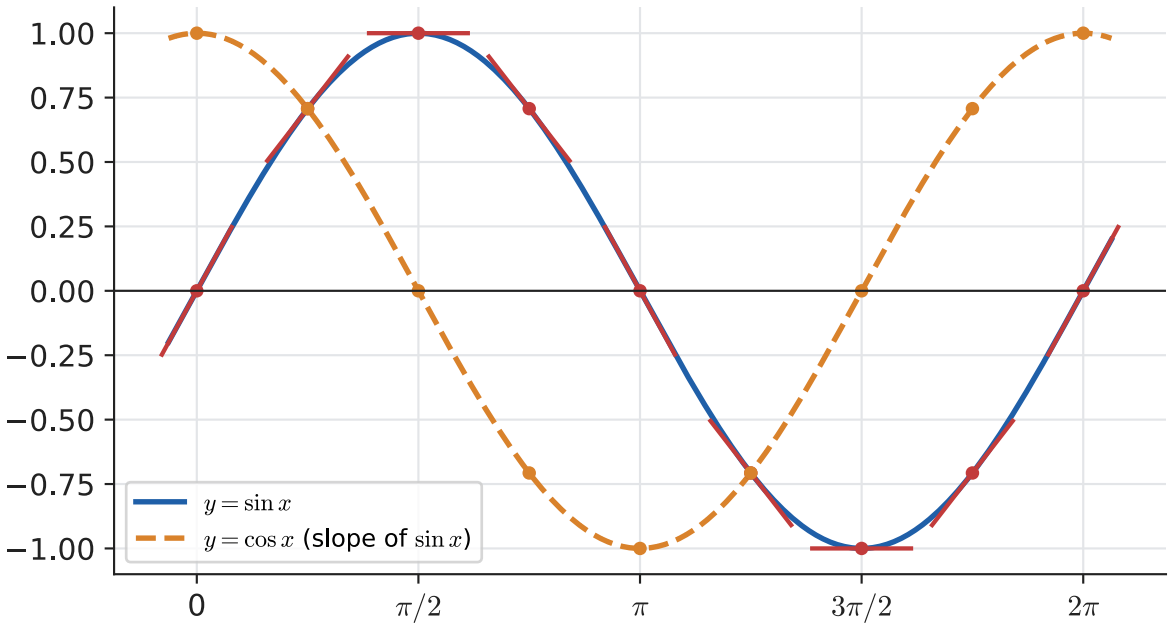


Figure 16. Tangent segments on $y = \sin x$: their slopes, plotted as dots, trace out $\cos x$.

Why it works. We need two limits established in Chapter 16 (with h in radians): *Try it:* Build cos x from slopes .

$$\lim_{h\to 0}\frac{\sin h}{h} = 1, \qquad \lim_{h\to 0}\frac{\cos h - 1}{h} = 0,$$

and the addition formula $\sin(x+h) = \sin x\cos h + \cos x \sin h$ from Volume 1, Chapter 13 . Then

$$\begin{aligned}
\frac{d}{dx}\sin x &= \lim_{h\to 0}\frac{\sin x\cos h + \cos x\sin h - \sin x}{h} && \text{(addition formula)}\\
&= \lim_{h\to 0}\left(\sin x\cdot\frac{\cos h - 1}{h} + \cos x\cdot\frac{\sin h}{h}\right) && \text{(group the } \sin x \text{ terms)}\\
&= \sin x\cdot 0 + \cos x\cdot 1 && \text{(the two standard limits)}\\
&= \cos x.
\end{aligned}$$

The same computation with $\cos(x+h) = \cos x\cos h - \sin x\sin h$ gives

$$\frac{d}{dx}\cos x = \cos x\cdot 0 - \sin x\cdot 1 = -\sin x.$$

From these two and the quotient rule, $\tan x = \sin x/\cos x$ gives

$$\frac{d}{dx}\tan x = \frac{\cos x\cdot\cos x - \sin x\cdot(-\sin x)}{\cos^2 x} = \frac{\cos^2 x + \sin^2 x}{\cos^2 x} = \frac{1}{\cos^2 x} = \sec^2 x.$$



Radians, always

$\frac{d}{dx}\sin x = \cos x$ is true only when x is measured in **radians**. The limit $\frac{\sin h}{h}\to 1$ depends on it. If x is in degrees, $\sin(x^\circ) = \sin\left(\frac{\pi x}{180}\right)$ and the chain rule gives $\frac{d}{dx}\sin(x^\circ) = \frac{\pi}{180}\cos(x^\circ)$, an ugly constant that radians exist to eliminate. In calculus, and in every programming language's `math.sin`, angles are radians.

17.6.2. The exponential function e^x

The problem. What is the derivative of an exponential b^x? Start from the definition and use an exponent law:

$$\begin{aligned}\frac{d}{dx}b^x &= \lim_{h\to 0}\frac{b^{x+h}-b^x}{h}\\ &= \lim_{h\to 0}\frac{b^x\,b^h-b^x}{h} && (b^{x+h}=b^xb^h)\\ &= b^x\cdot\lim_{h\to 0}\frac{b^h-1}{h} && (b^x \text{ does not depend on } h).\end{aligned}$$

So the derivative of any exponential is *proportional to itself*. The constant of proportionality, call it $M(b)=\lim_{h\to 0}(b^h-1)/h$, is the slope of b^x at $x=0$. Numerically: *Try it:* Slope of 2^x at 0 .

h	$b=2$	$b=e\approx 2.71828$	$b=3$
0.1	0.71773	1.05171	1.16123
0.01	0.69556	1.00502	1.10467
0.001	0.69339	1.00050	1.09922
0.0001	0.69317	1.00005	1.09867

For base 2 the slope at 0 is about 0.693; for base 3 it is about 1.099. Somewhere between 2 and 3 is the base whose slope at 0 is *exactly* 1. That base is $e=2.71828\ldots$, and for it

$$\boxed{\frac{d}{dx}e^x=e^x}$$

This is arguably the best definition of e: *the base for which the exponential function is its own derivative.* The compound-interest definition $e=\lim_{n\to\infty}(1+1/n)^n$ from Volume 1, Chapter 12 picks out the same number; showing they agree is a nice exercise in limits.

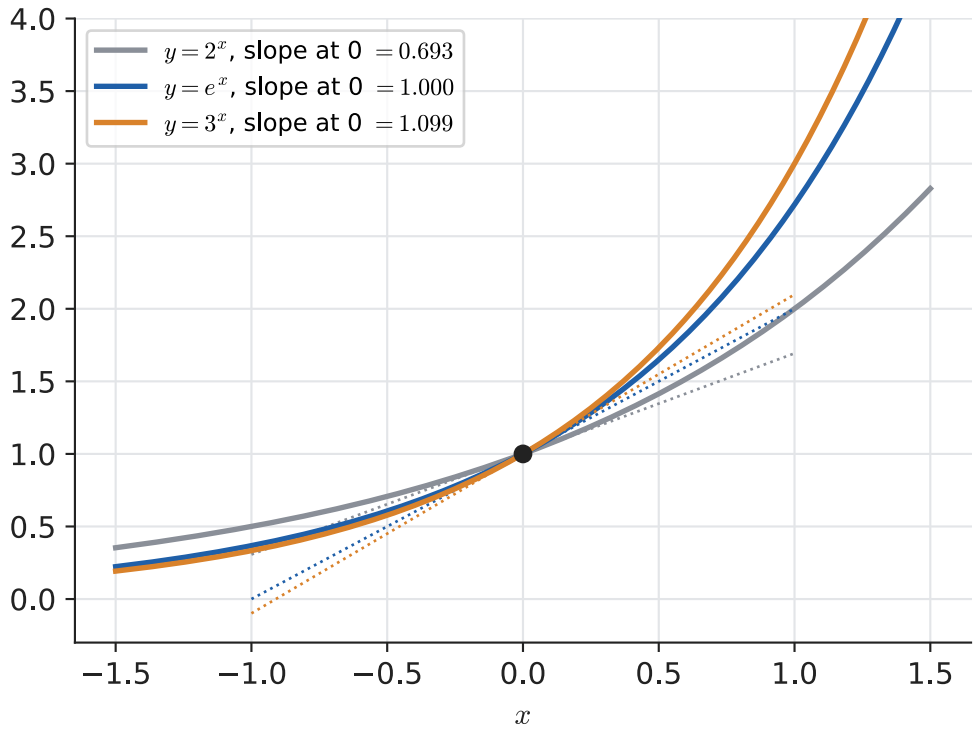


Figure 17. Exponentials 2^x, e^x, 3^x and their tangents at $x = 0$. Only e^x has slope exactly 1 there.

With the chain rule, $\frac{d}{dx}e^{kx} = k\,e^{kx}$ and $\frac{d}{dx}e^{g(x)} = g'(x)\,e^{g(x)}$. For example $\frac{d}{dx}e^{-x^2/2} = -x\,e^{-x^2/2}$: this is the bell curve of Volume 3, Chapter 30, and its slope formula tells you it rises for $x < 0$ and falls for $x > 0$. *Try it:* Slope of e^x at 0.

> *Why it matters that e^x is its own derivative*
>
> "Rate of growth proportional to current size" describes populations, compound interest, radioactive decay, cooling coffee, charging capacitors and unchecked epidemics. In symbols, $\frac{dy}{dt} = ky$. We have just shown $y = Ce^{kt}$ solves it, since $\frac{d}{dt}Ce^{kt} = kCe^{kt} = ky$. It turns out to be the *only* solution, which is why e turns up wherever things grow or decay. This is your first **differential equation**; Chapter 19 returns to them.

17.6.3. The natural logarithm $\ln x$

$\ln x$ is the inverse of e^x: $y = \ln x$ means exactly $e^y = x$. We know how to differentiate e^y, so differentiate both sides of $e^y = x$ with respect to x, treating y as a function of x and using the chain rule on the left:

$$\begin{aligned} e^y &= x \\ e^y \cdot \frac{dy}{dx} &= 1 \qquad && \text{(chain rule on the left)} \\ \frac{dy}{dx} &= \frac{1}{e^y} = \frac{1}{x} \qquad && (e^y = x). \end{aligned}$$

$$\boxed{\frac{d}{dx}\ln x = \frac{1}{x}} \quad (x > 0)$$

Geometrically: the graph of $\ln$ is the graph of $\exp$ reflected in the line $y = x$. Reflection swaps rise and run, so slopes become reciprocals. Where e^y has slope $e^y = x$, $\ln x$ has slope $1/x$.

Two dividends follow immediately.

Any base. Since $b = e^{\ln b}$, we have $b^x = e^{x \ln b}$, and the chain rule gives

$$\frac{d}{dx} b^x = \ln b \cdot e^{x \ln b} = (\ln b)\, b^x.$$

So the mysterious constant $M(b)$ in the table was $\ln b$ all along: $\ln 2 = 0.6931\ldots$, $\ln 3 = 1.0986\ldots$. Similarly $\frac{d}{dx} \log_b x = \frac{1}{x \ln b}$.

The power rule for every real exponent. For $x > 0$, $x^n = e^{n \ln x}$, so

$$\frac{d}{dx} x^n = e^{n \ln x} \cdot \frac{n}{x} = x^n \cdot \frac{n}{x} = n x^{n-1}.$$

This closes the gap left earlier: $\frac{d}{dx} x^\pi = \pi x^{\pi - 1}$ is now proved.



Standard derivatives

$f(x)$	$f'(x)$	$f(x)$	$f'(x)$
x^n	$n x^{n-1}$	e^x	e^x
$\sin x$	$\cos x$	b^x	$(\ln b)\, b^x$
$\cos x$	$-\sin x$	$\ln x$	$1/x$
$\tan x$	$\sec^2 x$	$\arctan x$	$1/(1+x^2)$

(Angles in radians. The $\arctan$ entry is derived in the next section.)

17.6.4. Where you'd use this

- **Everyday: cooling coffee.** A cup at 90 °C in a 20 °C room cools roughly as $T(t) = 20 + 70e^{-0.1t}$ (t in minutes). Then $T'(t) = -7e^{-0.1t}$: it is losing 7 °C per minute when poured, but only $7/e \approx 2.6$ °C per minute ten minutes later. The same formula shape describes a drug washing out of the bloodstream and a capacitor discharging.
- **Engineering: mains electricity.** UK mains voltage is $V(t) = 325 \sin(100\pi t)$ volts (230 V rms, 50 Hz). Its rate of change is $V'(t) = 325 \cdot 100\pi \cos(100\pi t)$, peaking at about $102{,}000$ volts per second as the voltage crosses zero. Capacitors pass current in proportion to $\frac{dV}{dt}$, which is why they block steady DC but conduct AC.
- **Finance.** £1000 growing at 5% compounded continuously is $A(t) = 1000e^{0.05t}$, and $A'(t) = 0.05A(t)$: the balance grows at £50 a year at the start and faster as it grows. "Rate proportional to size" is exactly $\frac{d}{dt} e^{kt} = ke^{kt}$.

- **Computing: the sigmoid and log-loss.** The logistic function $\sigma(x) = 1/(1+e^{-x})$, used to turn a score into a probability in logistic regression and neural networks, has the famously convenient derivative $\sigma'(x) = \sigma(x)(1-\sigma(x))$ (a chain-rule exercise). At $x = 0$, $\sigma = 0.5$ and the slope is 0.25. The log in the cross-entropy loss contributes the $\frac{1}{x}$ from $\frac{d}{dx}\ln x$; together they make the gradient of the loss with respect to the score simply "prediction minus truth".

17.7. Implicit differentiation

17.7.1. The problem

The circle $x^2 + y^2 = 25$ has a perfectly good tangent at $(3, 4)$. What is its slope? The trouble is that the circle is not the graph of a single function: solving gives $y = \pm\sqrt{25 - x^2}$, two functions glued together. For curves like $x^3 + y^3 = 6xy$ there is no convenient formula for y at all.

17.7.2. The idea

We don't need a formula for y. Near any point where the curve is smooth, it *locally* looks like the graph of some function $y(x)$, even if we can't write it down. So differentiate the equation as it stands, treating y as an unknown function of x. Each time you differentiate an expression in y, the chain rule produces a factor $\frac{dy}{dx}$; for example $\frac{d}{dx}y^2 = 2y\frac{dy}{dx}$. Then solve the resulting equation for $\frac{dy}{dx}$.



Implicit and explicit

A function is given **explicitly** when y is isolated: $y = \sqrt{25 - x^2}$. It is given **implicitly** by an equation relating x and y without isolating either: $x^2 + y^2 = 25$. **Implicit differentiation** finds dy/dx directly from the implicit equation.

Example 7. Procedure — implicit differentiation

Input: an equation in x and y; optionally a point (x_0, y_0) on the curve.

Steps:

1. Differentiate both sides with respect to x. Treat y as a function of x: every term containing y gets a factor $y' = \frac{dy}{dx}$ from the chain rule, and products like xy need the product rule.
2. Collect all terms containing y' on one side, everything else on the other.
3. Factor out y' and divide.
4. If a point was given, substitute it (check first that it really lies on the curve).

Output: y' as an expression in x and y, or its value at the point.

17.7.3. Worked example: the circle

$$\begin{aligned} \frac{d}{dx}(x^2 + y^2) &= \frac{d}{dx}(25) \\ 2x + 2y\,y' &= 0 && \left(\tfrac{d}{dx}y^2 = 2y\,y' \text{ by the chain rule}\right) \\ y' &= -\frac{x}{y}. \end{aligned}$$

At $(3, 4)$: $y' = -\frac{3}{4}$. Check against geometry: the radius to $(3, 4)$ has slope $\frac{4}{3}$, and a tangent to a circle is perpendicular to the radius, so its slope is the negative reciprocal, $-\frac{3}{4}$. Correct. At $(3, -4)$ the same formula gives $+\frac{3}{4}$, automatically handling the lower half.

17.7.4. Worked example: the folium of Descartes

Find the slope of $x^3 + y^3 = 6xy$ at $(3, 3)$. First check the point: $27 + 27 = 54 = 6 \cdot 3 \cdot 3$. Good.

$$\begin{aligned} 3x^2 + 3y^2\,y' &= 6y + 6x\,y' && \text{(product rule on } 6xy) \\ 3y^2\,y' - 6x\,y' &= 6y - 3x^2 && (y' \text{ terms on the left}) \\ y'\,(3y^2 - 6x) &= 6y - 3x^2 && \text{(factor)} \\ y' &= \frac{6y - 3x^2}{3y^2 - 6x} = \frac{2y - x^2}{y^2 - 2x} && \text{(divide; cancel 3).} \end{aligned}$$

At $(3, 3)$: $y' = \dfrac{6 - 9}{9 - 6} = -1$.

17.7.5. Worked example: derivative of $\arctan x$

Inverse functions are the other big use. $y = \arctan x$ means $\tan y = x$ with $-\frac{\pi}{2} < y < \frac{\pi}{2}$: *Try it:* Differentiate arctan x .

$$\begin{aligned} \sec^2 y \cdot y' &= 1 && \text{(differentiate } \tan y = x) \\ y' &= \frac{1}{\sec^2 y} = \frac{1}{1 + \tan^2 y} && (\sec^2 = 1 + \tan^2) \\ &= \frac{1}{1 + x^2} && (\tan y = x). \end{aligned}$$

A neat surprise: a trig-flavoured function whose derivative is a plain rational function. It is why π appears in $\int_{-\infty}^{\infty} \frac{dx}{1+x^2} = \pi$.



Implicit differentiation needs a smooth point

The formula $y' = -x/y$ fails at $(5, 0)$, where the circle's tangent is vertical: the denominator is zero. Implicit differentiation assumes the curve is locally a function graph near the point; where the tangent is vertical or the curve crosses itself (the folium does, at the origin, where the formula gives $0/0$), the method's answer is undefined and you must look at the geometry. The theorem that says when it works is the **implicit function theorem**, which you'll meet if you continue into Chapter 20.

17.7.6. Where you'd use this

- **Everyday and physics: the bicycle pump.** Squeeze air slowly (at constant temperature) and Boyle's law says PV stays constant, say $PV = 100$ (kPa and litres). You never need P as a formula in t: differentiate the constraint with respect to time, $V\frac{dP}{dt} + P\frac{dV}{dt} = 0$. When $V = 0.5$ L, $P = 200$ kPa, and you are compressing at $\frac{dV}{dt} = -0.1$ L/s, the pressure is rising at $\frac{dP}{dt} = -\frac{P}{V}\frac{dV}{dt} = -\frac{200}{0.5}(-0.1) = 40$ kPa/s. This is the technique of **related rates**, worked in detail (ladders, balloons, radar) in Section 18.5.
- **Science: constraints.** Orbits, chemical equilibria, and any "these quantities are locked together by an equation" situation are handled the same way: differentiate the constraint, not a solved formula.
- **Computing: implicit shapes.** Games and ray tracers often describe surfaces implicitly (a sphere is $x^2 + y^2 + z^2 = r^2$; signed-distance-field renderers build whole scenes this way). The tangent and normal at a hit point, which drive lighting and how a ball bounces off a curved bumper, come from differentiating the implicit equation, exactly as for the circle above (the multivariable version, the gradient, is in Chapter 20).

17.7.7. Exercises

1. Find dy/dx for $x^2y + y^3 = 10$, and its value at $(3, 1)$.
2. Use logarithmic differentiation to find the derivative of $y = x^x$ for $x > 0$. (Take $\ln$ of both sides first.)
3. Find the slope of the ellipse $4x^2 + 9y^2 = 72$ at $(3, 2)$.

▼ **Answers**

1. Check: $9 + 1 = 10$. Differentiate: $2xy + x^2y' + 3y^2y' = 0$, so $y'(x^2 + 3y^2) = -2xy$ and $y' = \dfrac{-2xy}{x^2 + 3y^2}$. At $(3, 1)$: $\dfrac{-6}{9 + 3} = -\frac{1}{2}$.
2. $\ln y = x \ln x$. Differentiate: $\dfrac{y'}{y} = \ln x + x \cdot \dfrac{1}{x} = \ln x + 1$. So $y' = x^x(\ln x + 1)$. (Neither the power rule nor the exponential rule applies: the variable is in both base and exponent.)
3. Check: $36 + 36 = 72$. $8x + 18y\,y' = 0$, so $y' = -\dfrac{8x}{18y} = -\dfrac{4x}{9y}$. At $(3, 2)$: $-\dfrac{12}{18} = -\frac{2}{3}$.

17.8. Higher derivatives: velocity, acceleration and beyond

17.8.1. The problem

The derivative of a function is itself a function, so we can differentiate *it*. Does that mean anything? In mechanics it means a great deal. If $s(t)$ is position, then $s'(t)$ is velocity, and the rate of change of velocity is acceleration: the quantity in Newton's second law, $F = ma$.

Higher derivatives

The **second derivative** is the derivative of the derivative: $f''(x)$, or $\dfrac{d^2y}{dx^2}$ ("dee two y by dee x squared"), or $\ddot{s}$ in mechanics. The Leibniz form comes from applying the operator twice: $\frac{d}{dx}\left(\frac{dy}{dx}\right)$.

The nth derivative is written $f^{(n)}(x)$ (parentheses, so it is not mistaken for a power) or $\dfrac{d^ny}{dx^n}$.

In mechanics: **position** s, **velocity** $v = \dot{s} = \frac{ds}{dt}$, **acceleration** $a = \dot{v} = \ddot{s} = \frac{d^2s}{dt^2}$. The third derivative is called **jerk**.

17.8.2. Worked example: a ball thrown upwards

A ball is thrown straight up at 20 m/s from ground level. Taking $g = 9.8$ m/s², its height after t seconds is $s(t) = 20t - 4.9t^2$ metres.

$$\begin{aligned} s(t) &= 20t - 4.9t^2 && \text{(position, m)} \\ v(t) = s'(t) &= 20 - 9.8t && \text{(velocity, m/s)} \\ a(t) = s''(t) &= -9.8 && \text{(acceleration, m/s}^2\text{).} \end{aligned}$$

Read the story from the derivatives. Acceleration is constant and negative: gravity, pulling down at all times, including at the top. Velocity starts at 20 m/s and decreases steadily. The ball is highest when it stops rising, $v(t) = 0$: $t = 20/9.8 \approx 2.04$ s, at height $s(2.04) \approx 20.4$ m. Going the other way (from acceleration back to position) is integration, and in the Mechanics course you did it every time you used $s = ut + \frac{1}{2}at^2$: that formula is just "integrate constant acceleration twice".

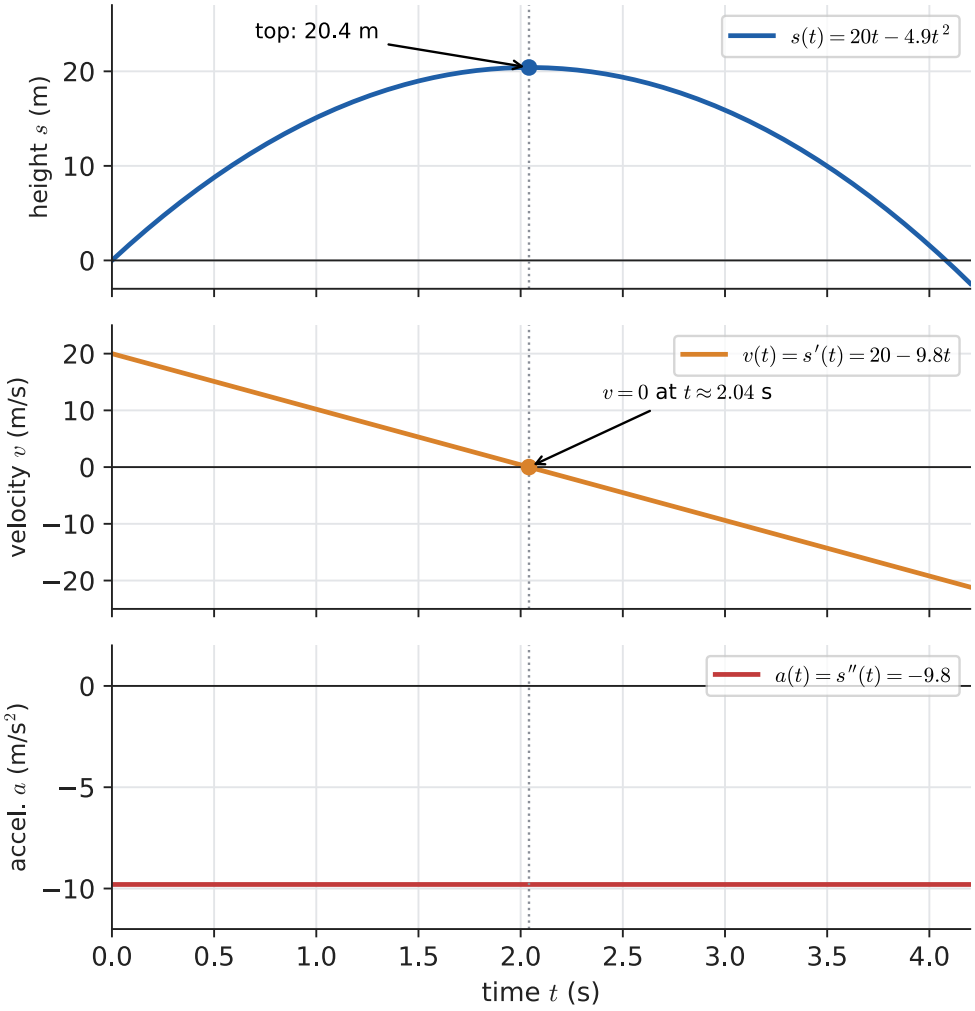


Figure 18. Position, velocity and acceleration of the ball. The top of the flight (upper panel) is exactly where velocity crosses zero (middle panel); acceleration (lower panel) is -9.8 throughout, including at the top.

The second derivative has a geometric meaning too: it is the rate at which the slope changes, so it measures **concavity**. $f'' > 0$ means slopes are increasing (the graph bends upwards, like a cup); $f'' < 0$ means it bends downwards. That is the backbone of the second-derivative test in Chapter 18.

A pattern for the curious. Differentiate $\sin x$ repeatedly: $\cos x, -\sin x, -\cos x, \sin x, \ldots$: period four. Differentiate x^4: $4x^3, 12x^2, 24x, 24, 0, 0, \ldots$: a polynomial of degree n dies after $n+1$ differentiations. Both facts are the seeds of Taylor series, previewed in Chapter 18.



Jerk and the smooth lift

Engineers care about the third derivative. A lift that switches instantly from zero to constant acceleration has infinite jerk at that moment, which you feel as a lurch. Lifts, trains, roller coasters, CNC machines and camera-motion paths in games are designed with bounded jerk (and sometimes bounded fourth derivative, whimsically called **snap**, followed by **crackle** and **pop**).

Where else second derivatives turn up. A news report saying "inflation is still rising, but more slowly" is a statement about a second derivative: prices are increasing ($P' > 0$) and the rate of increase is falling ($P'' < 0$). In a physics engine, Newton's second law $F = m\ddot{x}$ is turned into code by

computing acceleration from forces each frame and stepping velocity and position forward (Euler's method, Chapter 19). And in optimisation the second derivative measures curvature, telling Newton's method how big a step to take (Chapter 18).

17.9. When the derivative does not exist

17.9.1. The problem

Every rule so far assumed the limit in the definition exists. It doesn't always. Understanding *how* it can fail sharpens what the derivative means, and the failures appear in practice: the ReLU activation in neural networks has a corner, and absolute-value loss functions have corners exactly at their minimum. So does UK income tax: tax owed is a continuous function of income, but its slope (your marginal rate) jumps from 20% to 40% at the higher-rate threshold (£50,270 in recent tax years). Just below it, an extra £100 of pay costs £20 in income tax; just above, £40. The graph has a corner there, and "the" derivative at the threshold does not exist.

17.9.2. Three ways to fail

A corner: $f(x) = |x|$ at $x = 0$. The difference quotient is $\dfrac{|h| - 0}{h}$, which equals $+1$ for $h > 0$ and -1 for $h < 0$. The one-sided limits disagree ($+1$ from the right, -1 from the left), so the two-sided limit does not exist. Geometrically: secants from the right and secants from the left settle on two different lines. There is no single tangent.

A cusp: $f(x) = x^{2/3}$ at $x = 0$. The difference quotient is $\dfrac{h^{2/3}}{h} = h^{-1/3}$, which tends to $-\infty$ as $h \to 0^-$ and to ∞ (positive infinity) as h shrinks to 0 from the right. The graph comes into the origin vertically from both sides and turns back on itself.

A vertical tangent: $f(x) = x^{1/3}$ at $x = 0$. The difference quotient is $\dfrac{h^{1/3}}{h} = h^{-2/3} \to +\infty$ from both sides. There *is* a tangent line (the y-axis), but its slope is infinite, and the derivative must be a finite number.

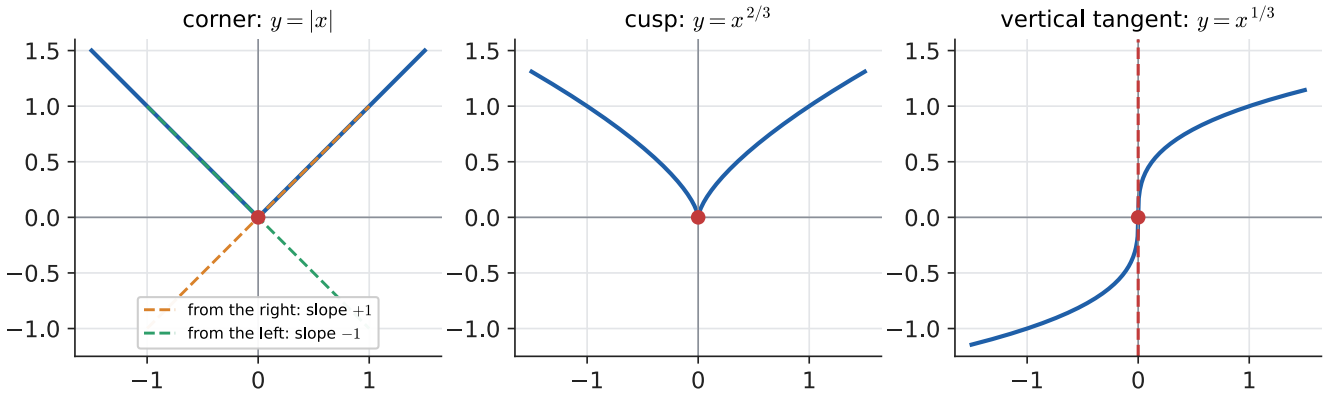


Figure 19. Three ways to fail to be differentiable at $x = 0$. At the corner, the dashed lines show the two different limiting secants: slope $+1$ from the right and -1 from the left.

A fourth: discontinuity. If f jumps at a, it cannot be differentiable there. In fact: *Try it:* Left and right slopes of |x| .

$$\textbf{differentiable at } a \implies \textbf{continuous at } a.$$

Proof (one line, and worth it): $f(a+h) - f(a) = \dfrac{f(a+h)-f(a)}{h} \cdot h \to f'(a) \cdot 0 = 0$ as $h \to 0$, so $f(a+h) \to f(a)$. The converse is false: $|x|$ is continuous everywhere but not differentiable at 0.



Continuous is not the same as smooth

"No breaks in the graph" (continuity) does not guarantee "a well-defined slope" (differentiability). When you differentiate a formula with the rules, you get an answer wherever the rules apply, and the rules can quietly hide the problem points: the power rule gives $\frac{d}{dx}x^{1/3} = \frac{1}{3}x^{-2/3}$, which is undefined at 0. That undefined value is the formula telling you something. Check piecewise definitions, absolute values, and fractional powers at the points where pieces meet or the formula blows up.



Continuous everywhere, differentiable nowhere

In 1872 Karl Weierstrass exhibited a function, $\sum_{n=0}^{\infty} a^n \cos(b^n \pi x)$ with suitable a, b, that is continuous everywhere but has a corner at *every* point. Mathematicians of the time called such functions "monsters". Today they're everyday objects: the path of a Brownian particle, and the stock-price models built on it, are continuous but nowhere differentiable. Zoom into a smooth curve and it straightens out; zoom into these and they stay jagged at every scale, like a fractal.

ReLU and subgradients

The most common activation function in deep learning, $\mathbf{ReLU}(x) = \max(0, x)$, has a corner at $x = 0$: derivative 0 to the left, 1 to the right, undefined at 0. Frameworks such as PyTorch simply *define* the derivative at 0 to be 0 and move on. That is justified by the idea of a **subgradient**: at a corner of a convex function, any slope between the left and right derivatives (here, anything in $[0, 1]$) is a valid "tangent" lying below the graph. Since the input is almost never exactly 0.0 in floating point, the choice rarely matters.

17.10. How computers compute derivatives

You have a function in code, not a formula on paper: a simulation, a rendering pipeline, a neural network with ten million parameters. You need its derivative. There are three approaches, and every working engineer should know which is which.

17.10.1. Numerical differentiation: just use a small h

The definition says the difference quotient approaches $f'(x)$ as $h \to 0$. So pick a small h and compute it. Two common formulas:

$$\textbf{forward difference: } \frac{f(x+h)-f(x)}{h}, \qquad \textbf{central difference: } \frac{f(x+h)-f(x-h)}{2h}.$$

The central difference uses a secant that straddles x symmetrically. Draw it on a parabola and you'll see why it's better: the errors on the two sides cancel. In fact for a quadratic the central difference is *exact* for any h.

How big is the error? Taylor polynomials (Chapter 18) tell us $f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots$. Substituting:

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \tfrac{h}{2}f''(x) + \cdots \qquad \textbf{(error proportional to } h\textbf{)}$$

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \tfrac{h^2}{6}f'''(x) + \cdots \qquad \textbf{(even powers cancel; error proportional to } h^2\textbf{)}.$$

This **truncation error** says: make h smaller, get a better answer. Shrinking h by 10 cuts the forward error by 10 and the central error by 100.

But computers don't do real arithmetic; they do floating point (Volume 1, Chapter 1), with about 16 significant digits ($\varepsilon \approx 2.2 \times 10^{-16}$). When h is tiny, $f(x+h)$ and $f(x)$ agree in most of their digits, and subtracting them (**catastrophic cancellation**) leaves only a few meaningful digits. Each value carries a rounding error of about $\varepsilon|f|$, and dividing by h amplifies it to about $\varepsilon|f|/h$. This **rounding error** grows as h shrinks.

Let's measure both. Differentiate $\sin$ at $x = 1$; the exact answer is $\cos 1$.

```python
import math

def forward_diff(f, x, h):
    return (f(x + h) - f(x)) / h

def central_diff(f, x, h):
    return (f(x + h) - f(x - h)) / (2 * h)

exact = math.cos(1.0)
print(f"{'h':>8} {'forward error':>15} {'central error':>15}")
for k in range(1, 13, 1):
    h = 10.0 ** -k
    ef = abs(forward_diff(math.sin, 1.0, h) - exact)
    ec = abs(central_diff(math.sin, 1.0, h) - exact)
    print(f"{h:8.0e} {ef:15.3e} {ec:15.3e}")
```

```
      h   forward error   central error
  1e-01       4.294e-02       9.001e-04
  1e-02       4.216e-03       9.005e-06
  1e-03       4.208e-04       9.005e-08
  1e-04       4.207e-05       9.004e-10
  1e-05       4.207e-06       1.114e-11
  1e-06       4.207e-07       2.772e-11
  1e-07       4.183e-08       1.943e-10
  1e-08       1.407e-08       2.970e-09
  1e-09       5.254e-08       2.970e-09
  1e-10       5.848e-08       5.848e-08
  1e-11       1.169e-06       1.169e-06
  1e-12       4.324e-05       1.227e-05
```

The theory predicts the top of the table exactly: forward error $\approx \frac{h}{2}|\sin 1| = 0.42h$, central error $\approx \frac{h^2}{6}\cos 1 = 0.090h^2$. Then the trend reverses. The forward difference is best around $h \approx 10^{-8} \approx \sqrt{\varepsilon}$, with about 8 correct digits; the central difference is best around $h \approx 10^{-5} \approx \varepsilon^{1/3}$, with about 11. Make h smaller than that and the answer gets *worse*.

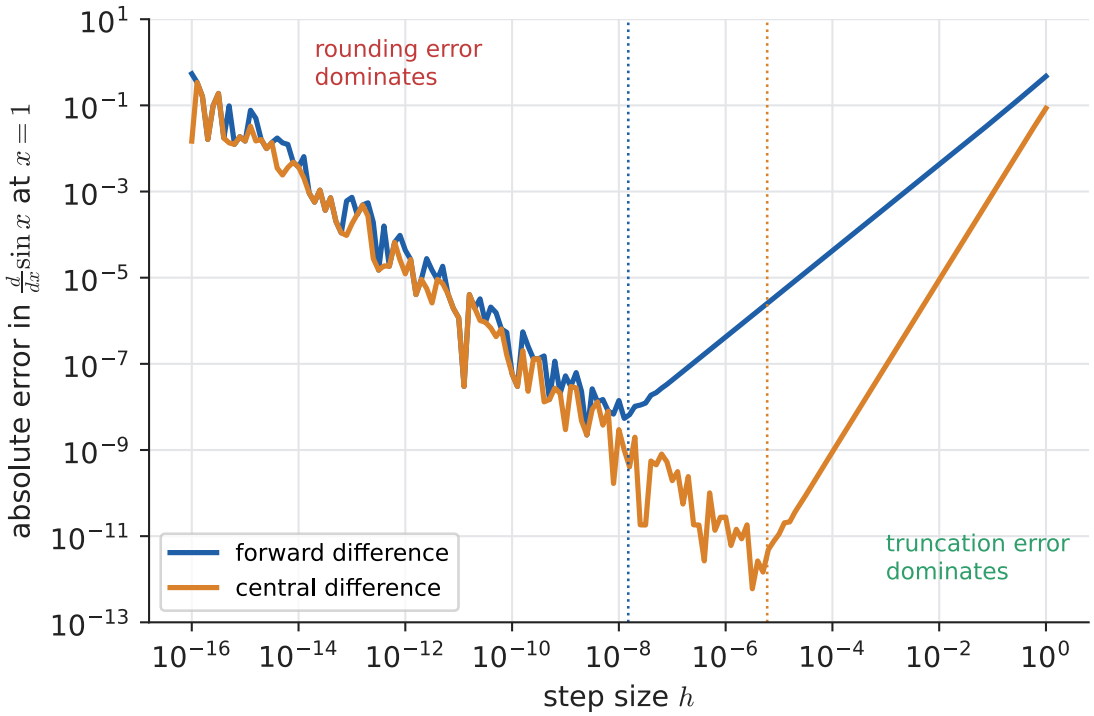


Figure 20. Total error of numerical differentiation. Truncation error falls as h shrinks (straight lines on the right); rounding error rises (noisy region on the left). Dotted lines mark the theoretical sweet spots.



Smaller is not always better

"Take the limit as $h \to 0$" is a mathematical instruction, not a numerical one. In floating point, $h = 10^{-15}$ gives a derivative with essentially no correct digits, and $h = 10^{-17}$ gives zero, because $1 + 10^{-17}$ rounds to exactly 1. Rules of thumb: forward difference $h \approx 10^{-8} \cdot \max(1, |x|)$; central difference $h \approx 10^{-5} \cdot \max(1, |x|)$. And numerical derivatives of noisy data (sensor readings) amplify the noise; smooth first.



Finding the sweet spot

A slider sets $\log_{10} h$ from -16 to 0 and plots forward and central difference errors for a function you choose ($\sin$, e^x, x^3, $\sqrt{x}$ near 0). Find the best h for each. Then switch to 32-bit floats and watch the sweet spot move to much larger h.

Try it: Find the sweet spot · The sweet spot in float32 · Find the sweet spot

17.10.2. Automatic differentiation: exact derivatives without formulas

Numerical differentiation is approximate and fragile. Symbolic differentiation (applying the rules to a formula tree, as SymPy does) is exact but needs a formula and can produce enormous expressions. There is a third way, and it powers essentially all modern machine learning: **automatic differentiation** (AD). It applies the rules of this chapter, not to formulas, but to the actual *numbers* flowing through your program. *Try it:* Dual numbers, node by node .

The forward-mode version rests on a small algebraic trick. Invent a number ϵ with $\epsilon \neq 0$ but $\epsilon^2 = 0$ (like $i^2 = -1$ in Volume 1, Chapter 15 , it's an extension of the number system defined by a rule). Numbers of the form $a + b\epsilon$ are called **dual numbers**. Multiply two:

$$(a + b\epsilon)(c + d\epsilon) = ac + (ad + bc)\epsilon + bd\,\epsilon^2 = ac + (ad + bc)\epsilon.$$

The ϵ part is the product rule: if b and d are the derivatives of two quantities a and c, then $ad + bc$ is the derivative of their product. More generally, for any polynomial f, expanding $f(x + \epsilon)$ gives $f(x) + f'(x)\epsilon$ exactly, because every ϵ^2 and higher term vanishes. Evaluate your function on $x + 1\epsilon$ and the derivative falls out in the ϵ slot, exact to floating-point precision, with no h to choose.

PYTHON

```python
import math

class Dual:
    """A number a + b*eps with eps**2 == 0. b carries the derivative."""
    def __init__(self, val, der=0.0):
        self.val, self.der = val, der

    def __add__(self, other):
        other = other if isinstance(other, Dual) else Dual(other)
        return Dual(self.val + other.val, self.der + other.der)
    __radd__ = __add__

    def __sub__(self, other):
        other = other if isinstance(other, Dual) else Dual(other)
        return Dual(self.val - other.val, self.der - other.der)

    def __mul__(self, other):
        other = other if isinstance(other, Dual) else Dual(other)
        # product rule lives here
        return Dual(self.val * other.val,
                    self.der * other.val + self.val * other.der)
    __rmul__ = __mul__

    def __truediv__(self, other):
        other = other if isinstance(other, Dual) else Dual(other)
        # quotient rule lives here
        return Dual(self.val / other.val,
                    (self.der * other.val - self.val * other.der) / other.val ** 2)

def sin(x):
    # chain rule: d/dx sin(u) = cos(u) * u'
    return Dual(math.sin(x.val), math.cos(x.val) * x.der)

def exp(x):
    return Dual(math.exp(x.val), math.exp(x.val) * x.der)

def derivative(f, x):
    return f(Dual(x, 1.0)).der

f = lambda x: x * x * x + 2 * x          # f'(x) = 3x^2 + 2
g = lambda x: sin(x * x)                  # g'(x) = 2x cos(x^2)
h = lambda x: exp(2 * x) / (1 + x * x)    # quotient + chain

print(derivative(f, 2.0))
print(derivative(g, 1.5), 2 * 1.5 * math.cos(1.5 ** 2))
x = 0.5
exact_h = (2 * math.exp(2*x) * (1 + x*x) - math.exp(2*x) * 2*x) / (1 + x*x)**2
print(derivative(h, 0.5), exact_h)
```

```
14.0
-1.8845208681682175 -1.8845208681682175
2.609550555320683 2.609550555320683
```

Look at what each method of the class contains: the sum rule, the product rule, the quotient rule, and (in `sin` and `exp`) the chain rule. The rules of this chapter are the whole implementation. The derivatives agree with the hand-derived formulas to every printed digit.

17.10.3. Backpropagation is the chain rule

Forward mode pushes one derivative through the program alongside the values. That costs one pass per *input* variable. A neural network has millions of inputs (its weights) and one output (the loss), so forward mode would need millions of passes. **Reverse mode** turns it around: run the program forward once, recording intermediate values, then walk backwards multiplying local derivatives via the chain rule. One backward pass gives the derivative of the output with respect to *every* input. In machine learning, reverse-mode AD is called **backpropagation**. *Try it:* Backpropagate the one-weight network .

Here is a one-weight "network": input x, weight w, prediction $a = \tanh(wx)$, loss $L = (a - \text{target})^2$. As a chain: $w \to z = wx \to a = \tanh z \to L$. The chain rule says

$$\frac{dL}{dw} = \frac{dL}{da} \cdot \frac{da}{dz} \cdot \frac{dz}{dw} = 2(a - \text{target}) \cdot (1 - a^2) \cdot x,$$

using $\frac{d}{dz}\tanh z = 1 - \tanh^2 z$.

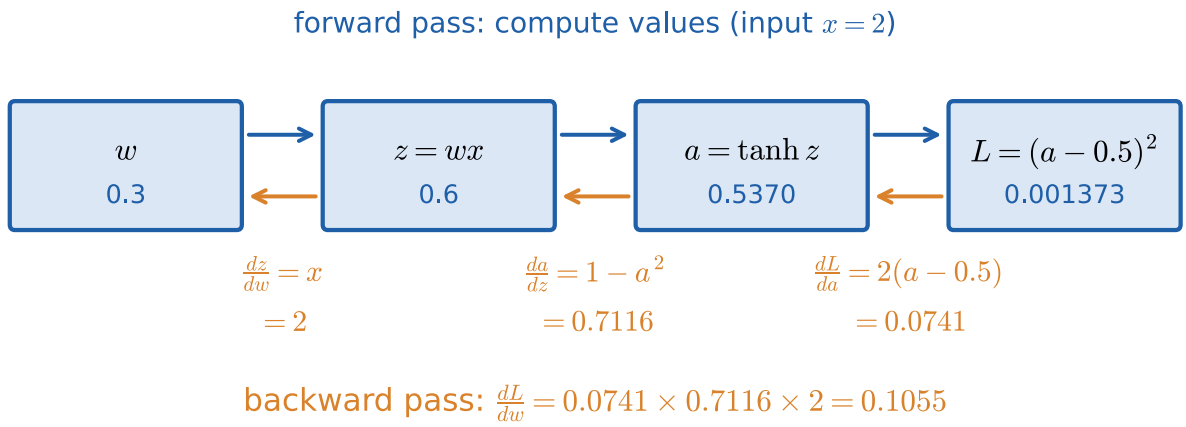


Figure 21. Backpropagation on the one-weight network with $w = 0.3$, $x = 2$, target 0.5. The forward pass (blue) computes and stores each value; the backward pass (orange) multiplies the local derivatives on each link, from the loss back to the weight.

PYTHON

```python
import math

def loss(w, x=2.0, target=0.5):
    z = w * x                 # stage 1
    a = math.tanh(z)          # stage 2
    return (a - target) ** 2  # stage 3

def grad_loss(w, x=2.0, target=0.5):
    # forward pass: remember intermediate values
    z = w * x
    a = math.tanh(z)
    L = (a - target) ** 2
    # backward pass: multiply local derivatives, last stage first
    dL_da = 2 * (a - target)
    da_dz = 1 - a ** 2        # derivative of tanh
    dz_dw = x
    return L, dL_da * da_dz * dz_dw

w = 0.3
L, g = grad_loss(w)
numeric = (loss(w + 1e-6) - loss(w - 1e-6)) / 2e-6
print(f"loss = {L:.6f}")
print(f"chain-rule gradient = {g:.6f}")
print(f"central-difference  = {numeric:.6f}")
```

```
loss = 0.001373
chain-rule gradient = 0.105455
central-difference  = 0.105455
```

The central difference is used here the way practitioners use it: as a **gradient check**, an independent test that the hand-written backward pass is right. A real network has millions of weights and many layers, so the chain has many links and branches, but every step of backpropagation is this: a local derivative, multiplied into the running product, from the output back towards the inputs. Gradient descent (Chapter 18, Chapter 20, Volume 3, Chapter 33) then nudges each weight downhill.

Computer-science connection: three ways to differentiate a program

Method	Strengths	Weaknesses
Numerical (finite differences)	Works on any black box; trivial to code	Approximate; h must be tuned; one or two evaluations per input variable; fails on noisy functions
Symbolic (SymPy, Mathematica)	Exact formula you can read and reuse	Needs a closed-form expression; "expression swell"; can't handle loops and branches naturally
Automatic (JAX, PyTorch autograd)	Exact to rounding; works through loops and branches; reverse mode costs a small constant multiple of one function evaluation, whatever the number of inputs	Needs the code written in a differentiable framework; reverse mode stores intermediates (memory)

Method selection: use AD for anything you will optimise (ML, simulation tuning); symbolic for deriving formulas you will reason about; finite differences for quick checks and for testing your AD or hand-written gradients.



Why it took until the 2010s

Reverse-mode automatic differentiation was described by Seppo Linnainmaa in his 1970 master's thesis, and backpropagation for neural networks was popularised by Rumelhart, Hinton and Williams in 1986. The mathematics, the chain rule, is from Leibniz's notebooks of the 1670s. What changed in the 2010s was data and GPUs; the calculus was already three centuries old.



Build a computation graph

Drag nodes ($+$, $\times$, $\sin$, $\exp$, x^2) onto a canvas to build an expression graph. Press "forward" to watch values flow from inputs to output; press "backward" to watch local derivatives multiply their way back from the output, with each edge labelled by its local derivative. Compare the result with forward-mode dual numbers and with a finite difference.

Try it: Build a computation graph

17.11. Chapter exercises

1. **(Recognition)** Which of these is the definition of $f'(2)$? (a) $\dfrac{f(2+h)-f(2)}{h}$; (b) $\lim_{h\to 0}\dfrac{f(2+h)-f(2)}{h}$; (c) $\lim_{h\to 0}\dfrac{f(h)-f(2)}{h}$.

2. **(Recognition)** For each, name the rule(s) you would use first: $e^{x}\cos x$; $\cos(e^{x})$; $\dfrac{e^{x}}{\cos x}$; $e^{x}+\cos x$.

3. **(Application)** Differentiate: (a) $x^{3}e^{2x}$; (b) $\ln(\cos x)$; (c) $\dfrac{\sin x}{1+\cos x}$; (d) $e^{\sin 3x}$.

4. **(Application)** A particle moves with $s(t)=t^{3}-9t^{2}+24t$ metres. When is it momentarily at rest? What is its acceleration at those times?

5. **(Problem solving)** Find the equation of the tangent line to $y=x\ln x$ at $x=e$.

6. **(Problem solving)** For what value of k is the piecewise function $f(x)=x^{2}$ for $x\le 1$, $f(x)=kx-1$ for $x>1$ both continuous and differentiable at $x=1$?

7. **(Problem solving, CS)** Extend the `Dual` class with a `cos` function and a *pow* method for a constant integer exponent. Test on $\cos(x^{3})$ at $x=1$.

▼ **Answers**

1. (b). Option (a) has no limit, so it is a secant slope; (c) uses the wrong numerator.

2. Product rule; chain rule; quotient rule; sum rule.

3. (a) Product + chain: $3x^{2}e^{2x}+x^{3}\cdot 2e^{2x}=x^{2}e^{2x}(3+2x)$. (b) Chain: $\dfrac{1}{\cos x}\cdot(-\sin x)=-\tan x$. (c) Quotient: $\dfrac{\cos x(1+\cos x)-\sin x(-\sin x)}{(1+\cos x)^{2}}=\dfrac{\cos x+\cos^{2}x+\sin^{2}x}{(1+\cos x)^{2}}=\dfrac{1+\cos x}{(1+\cos x)^{2}}=\dfrac{1}{1+\cos x}$. (d) Chain, two layers: $e^{\sin 3x}\cdot\cos 3x\cdot 3=3\cos(3x)\,e^{\sin 3x}$.

4. $v=3t^{2}-18t+24=3(t^{2}-6t+8)=3(t-2)(t-4)$, so at rest at $t=2$ and $t=4$ s. $a=6t-18$: $a(2)=-6$ m/s² (slowing to a stop then reversing), $a(4)=6$ m/s².

5. $y'=1\cdot\ln x+x\cdot\frac{1}{x}=\ln x+1$. At $x=e$: $y=e\ln e=e$, slope $=1+1=2$. Tangent: $y-e=2(x-e)$, i.e. $y=2x-e$.

6. Continuity at 1: $1^{2}=k\cdot 1-1$, so $k=2$. Differentiability: left derivative $2x=2$ at $x=1$; right derivative $k=2$. They match, so $k=2$ gives both.

7. One possible solution:

```python
def cos(x):
    return Dual(math.cos(x.val), -math.sin(x.val) * x.der)

def _pow(self, n):           # n a plain int
    return Dual(self.val ** n, n * self.val ** (n - 1) * self.der)
Dual.__pow__ = _pow

print(derivative(lambda x: cos(x ** 3), 1.0))   # -2.5244129544236893
print(-3 * math.sin(1.0))                         # same: -3 x^2 sin(x^3) at 1
```

17.12. Interactive tools

Browser tools on the companion website that go with this chapter. Each example link in the chapter opens one of them with that example already set up.

- Calculus, visually
- Graphing calculator
- Symbolic algebra
- Where floating point bites
- The maths of machine learning

17.13. Mastery check

1. Using the definition, find $f'(x)$ for $f(x) = 3x^2 - x$.
2. Differentiate $y = x^3 e^x$.
3. Differentiate $y = \ln(x^2 + 1)$.
4. Differentiate $y = \dfrac{\sin x}{x}$.
5. Find dy/dx at $(1, 2)$ on the curve $x^2 + xy + y^2 = 7$.
6. Where is $f(x) = |x - 2|$ not differentiable, and why?
7. A particle has position $s(t) = t^3 - 6t^2 + 9t$. Find its velocity and acceleration at $t = 2$.
8. You estimate $\frac{d}{dx} e^x$ at $x = 0$ with a forward difference and $h = 10^{-3}$. Roughly how large is the error, and how would you reduce it?

Decision rule: All correct without hesitation → move on to Chapter 18. Stuck on 1 → reread Section 17.4. Stuck on 2–4 → reread Section 17.5 and Section 17.6 and do their exercises. Stuck on 5 → Section 17.7. Stuck on 6 → Section 17.9. Stuck on 8 → Section 17.10.

▼ **Mastery check answers**

1. $\dfrac{3(x+h)^2 - (x+h) - 3x^2 + x}{h} = \dfrac{6xh + 3h^2 - h}{h} = 6x + 3h - 1 \to 6x - 1.$

2. Product rule: $3x^2e^x + x^3e^x = x^2(x+3)e^x$.
3. Chain rule: $\dfrac{1}{x^2+1}\cdot 2x = \dfrac{2x}{x^2+1}$.
4. Quotient rule: $\dfrac{x\cos x - \sin x}{x^2}$.
5. Check: $1+2+4=7$. Differentiate: $2x + (y + x\,y') + 2y\,y' = 0$, so $y'(x+2y) = -(2x+y)$, $y' = -\dfrac{2x+y}{x+2y}$. At $(1,2)$: $-\dfrac{4}{5}$.
6. At $x = 2$ only. It has a corner there: the left derivative is -1 and the right derivative is $+1$, so the limit of the difference quotient does not exist. It is still continuous there.
7. $v = 3t^2 - 12t + 9$, so $v(2) = 12 - 24 + 9 = -3$ m/s (moving backwards). $a = 6t - 12$, so $a(2) = 0$.
8. Truncation error $\approx \frac{h}{2}f''(0) = \frac{10^{-3}}{2}\cdot 1 = 5\times 10^{-4}$; indeed $(e^{0.001}-1)/0.001 = 1.0005002$. Reduce it by switching to a central difference (error $\approx h^2/6 \approx 1.7\times 10^{-7}$ at the same h), or by decreasing h towards 10^{-8} but no further (rounding error takes over), or by using automatic differentiation, which is exact.

US curriculum

AP Calculus AB covers everything in this chapter except the computing section: the limit definition, all the differentiation rules, implicit differentiation, derivatives of inverse functions, higher derivatives and motion problems, and the relationship between differentiability and continuity. **AP Calculus BC** adds parametric and polar derivatives. Numerical differentiation appears in college **Numerical Analysis**; automatic differentiation is a graduate-level or machine-learning topic, though a strong high-school programmer can follow the dual-number code above.

Chapter 18. Using Derivatives: Optimisation, Approximation, and Newton's Method

In Chapter 17 you built the derivative: the instantaneous rate of change, the slope of the tangent line, and a toolkit of rules for computing it. This chapter puts that machine to work. A single number, $f'(x)$, turns out to tell you where a function rises and falls, where it peaks, how to build the best box from a sheet of card, how fast a ladder's top slides down a wall, how to approximate a hard function by an easy one, and — the chapter's centrepiece — how to solve equations that algebra cannot touch, using Newton's method. Throughout, the same idea recurs: near any point, a smooth curve looks like its tangent line, and lines are easy.

Why this chapter matters: this is where calculus starts paying rent. Setting a price, sizing a checkpoint interval, dosing a drug, judging how a measurement error spreads through a calculation, computing a square root inside a chip, and training a machine-learning model are all applications of the handful of ideas below.

Where this chapter sits

You need: derivatives and the rules for computing them (Chapter 17), limits (Chapter 16), graphs of functions (Volume 1, Chapter 11), solving polynomial and quadratic equations (Volume 1, Chapter 7 , Volume 1, Chapter 8), the exponential, logarithm and trigonometric functions (Volume 1, Chapter 12 , Volume 1, Chapter 13), and sequences (Volume 1, Chapter 14) for the iteration in Newton's method.

This chapter unlocks: the integral, which reverses the derivative (Chapter 19); optimisation in many variables and the gradient (Chapter 20); numerical root-finding and floating-point error analysis (Chapter 32); and gradient-based machine learning (Volume 3, Chapter 33).

Fast path: if you can find and classify the stationary points of $x^3 - 3x^2 - 9x + 5$ and do two Newton steps for $\sqrt{2}$ from $x_0 = 1$ by hand, skim to Section 18.10.11 and the mastery check.

18.1. What the derivative says about shape

18.1.1. The problem

You are handed a formula — say the profit of a product as a function of its price, or the height of a projectile as a function of time — and asked: *where is it largest?* Plotting helps, but a plot is only as good as the points you sampled, and it never *proves* anything. You want a way to read the shape of a graph directly from the formula.

The derivative is exactly that tool. Recall that $f'(x)$ is the slope of the tangent line at x. Slope has a sign, and the sign is a direction:

- $f'(x) > 0$: the tangent tilts upward, so the graph is climbing;

- $f'(x) < 0$: the tangent tilts downward, so the graph is falling;
- $f'(x) = 0$: the tangent is horizontal — the graph is momentarily level.

18.1.2. Intuition

Walk along a hilly road from left to right. While you climb, your altimeter's rate of change is positive. At the very top of a hill — for one instant — you are neither climbing nor descending: the rate is zero. Then you start to descend and the rate turns negative. So the summit is betrayed by the rate *passing through zero and changing sign from + to -*. A valley floor is the reverse: − then 0 then +.

That is the whole idea of this section. Peaks and troughs hide at the places where f' is zero (or fails to exist), and the *sign change* of f' around those places tells you which is which.

18.1.3. Why it works

Why must a smooth function with a positive derivative everywhere on an interval actually be increasing there? Positive slope at every single point *feels* like it should force the function up, but "slope at a point" is a limit — an instantaneous statement — and "increasing" compares two different points.

The bridge is the **Mean Value Theorem**: if f is differentiable on an interval and $a < b$ are two points in it, then somewhere between them there is a point c where the tangent slope equals the average slope:

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Picture it: draw the chord (secant) from $(a, f(a))$ to $(b, f(b))$; slide a parallel line toward the curve, and it must touch the curve somewhere as a tangent. Now rearrange: $f(b) - f(a) = f'(c)\,(b - a)$. If f' is positive everywhere and $b - a > 0$, the right-hand side is positive, so $f(b) > f(a)$. The function is increasing. Same argument with signs flipped for decreasing, and with $f' = 0$ everywhere it says $f(b) = f(a)$: a function with zero derivative is constant. (That last fact is the reason antiderivatives come with a "+ C" in Chapter 19.)

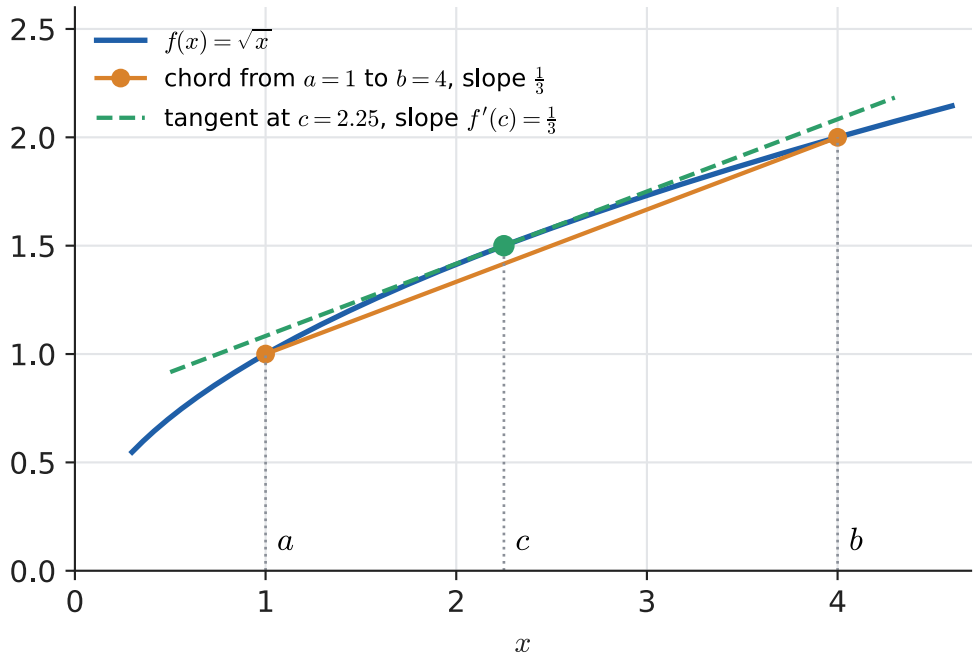


Figure 22. The Mean Value Theorem for $f(x) = \sqrt{x}$ on $[1, 4]$. The chord has slope $\frac{2-1}{4-1} = \frac{1}{3}$; at $c = 2.25$ the tangent (dashed) has exactly that slope, since $f'(c) = \frac{1}{2\sqrt{2.25}} = \frac{1}{3}$.

The Mean Value Theorem is also the legal basis of average-speed cameras: if your average speed between two cameras was 62 mph, the theorem guarantees that at some instant your speedometer read exactly 62 mph. Over a 50 mph limit, that is enough for a ticket. *Try it:* Mean value point for √x .

18.1.4. Vocabulary and notation

Increasing and decreasing

A function f is **increasing** on an interval if $a < b$ implies $f(a) < f(b)$ for any two points in the interval, and **decreasing** if $a < b$ implies $f(a) > f(b)$. Test: $f' > 0$ on an interval ⇒ increasing there; $f' < 0$ ⇒ decreasing.

Critical point (stationary point)

A **critical point** of f is a point c in its domain where $f'(c) = 0$ or $f'(c)$ does not exist. Where $f'(c) = 0$ specifically, UK texts say **stationary point**: the tangent is horizontal. Every local maximum or minimum of a function at an interior point is a critical point — but not every critical point is a maximum or minimum.

Local and global extrema

f has a **local maximum** at c if $f(c) \geq f(x)$ for all x *near* c; a **local minimum** if $f(c) \leq f(x)$ nearby. A **global** (or **absolute**) maximum or minimum compares against *every* point of the domain. **Extremum** (plural **extrema**) means "maximum or minimum". UK texts often say **turning point** for a local extremum.



Concave up, concave down, inflection point

f is **concave up** (UK: **convex**) on an interval where its slope is increasing — the graph bends like a cup, ∪. It is **concave down** (UK: **concave**) where its slope is decreasing — a cap, ∩. Test: $f'' > 0 \Rightarrow$ concave up; $f'' < 0 \Rightarrow$ concave down. An **inflection point** is a point on the graph where the concavity changes.

The UK/US clash in the concavity vocabulary is real and causes confusion: a US textbook's "concave up" is a British "convex". In optimisation and machine learning the British/European usage wins: a **convex function** is cup-shaped, and convexity guarantees that any local minimum is a global one.

18.1.5. The method: first and second derivative tests

Example 8. Procedure — First derivative test

Input: a differentiable function f (except possibly at isolated points).

Steps:

1. Compute $f'(x)$ and factor it as far as possible.
2. Find the critical points: solve $f'(x) = 0$, and list points where f' is undefined but f is defined.
3. Mark the critical points on a number line. They split it into intervals.
4. In each interval pick a test point and find the sign of f' there (factored form makes this a sign-counting exercise).
5. Classify each critical point c:
   a. sign changes $+ \to -$: local maximum;
   b. sign changes $- \to +$: local minimum;
   c. no sign change: neither (a level spot on a rising or falling stretch).

Output: the intervals of increase and decrease, and the classification of every critical point.

The second derivative gives a quicker test at a stationary point c (where $f'(c) = 0$):

- $f''(c) > 0$: concave up at c — a cup — so c is a local **minimum**;
- $f''(c) < 0$: concave down — a cap — so c is a local **maximum**;
- $f''(c) = 0$: **the test says nothing**. Fall back on the first derivative test.

18.1.6. Worked example: a cubic

Find and classify the critical points of $f(x) = x^3 - 3x^2 - 9x + 5$, find where it is concave up and down, and locate its inflection point. *Try it:* Differentiate the cubic .

Step 1 — differentiate and factor.

$$\begin{aligned} f'(x) &= 3x^2 - 6x - 9 && \text{(power rule, term by term)} \\ &= 3(x^2 - 2x - 3) && \text{(take out the common factor 3)} \\ &= 3(x+1)(x-3) && \text{(two numbers with product } -3\text{, sum } -2\text{: } +1, -3) \end{aligned}$$

Step 2 — critical points. $f'(x) = 0$ when $x = -1$ or $x = 3$. f' is a polynomial, defined everywhere, so there are no others.

Step 3 — sign chart. The critical points split the line into three intervals. Test one point in each:

Interval	test x	$x+1$	$x-3$	sign of $f'(x) = 3(x+1)(x-3)$
$x < -1$	-2	$-$	$-$	$+$ (increasing)
$-1 < x < 3$	0	$+$	$-$	$-$ (decreasing)
$x > 3$	4	$+$	$+$	$+$ (increasing)

Step 4 — classify and evaluate.

$$\begin{aligned} f(-1) &= (-1)^3 - 3(-1)^2 - 9(-1) + 5 = -1 - 3 + 9 + 5 = 10 && (+ \to -\text{: local max}) \\ f(3) &= 3^3 - 3(3)^2 - 9(3) + 5 = 27 - 27 - 27 + 5 = -22 && (- \to +\text{: local min}) \end{aligned}$$

Step 5 — check with the second derivative. $f''(x) = 6x - 6$.

$$\begin{aligned} f''(-1) &= -12 < 0 && \text{(concave down: max, agrees)} \\ f''(3) &= 12 > 0 && \text{(concave up: min, agrees)} \end{aligned}$$

Step 6 — concavity and inflection. $f''(x) = 6(x-1)$ is negative for $x < 1$ and positive for $x > 1$. The concavity changes at $x = 1$, where $f(1) = 1 - 3 - 9 + 5 = -6$. The inflection point is $(1, -6)$ — exactly halfway between the two turning points, as it always is for a cubic.

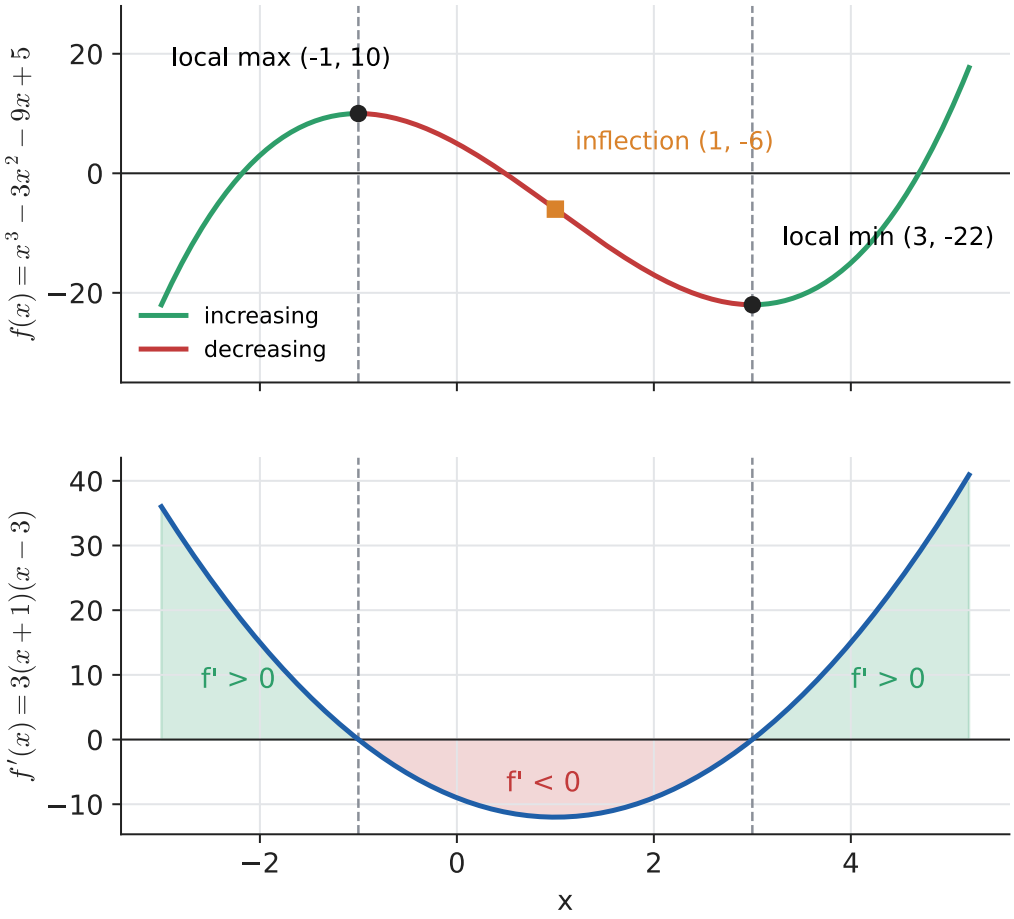


Figure 23. A cubic and its derivative. Where $f' > 0$ (green), f rises; where $f' < 0$ (red), it falls. The turning points of f sit directly above the zeros of f'.

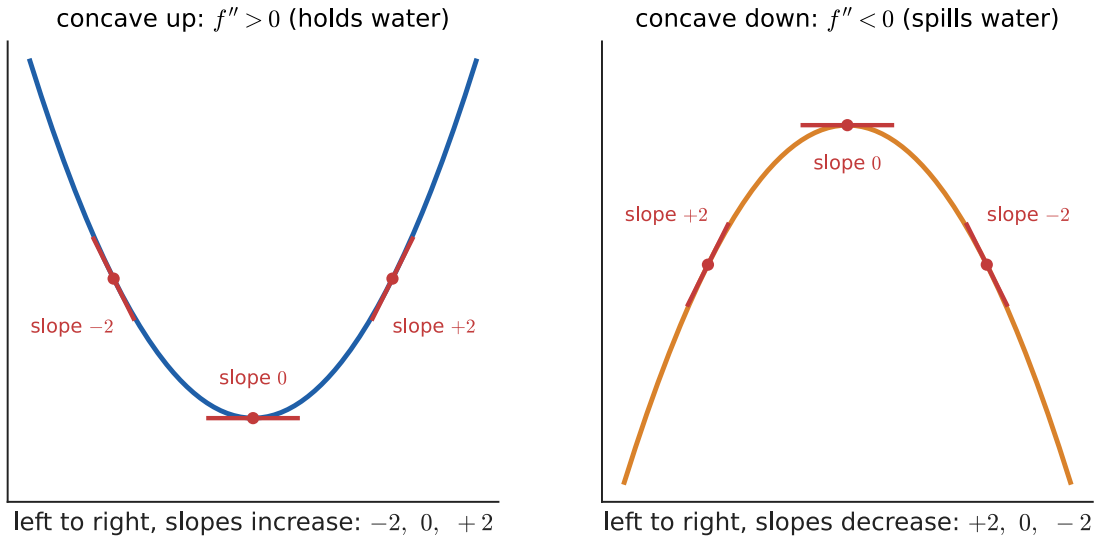


Figure 24. Concavity: in a cup the tangent slopes increase from left to right ($f'' > 0$); in a cap they decrease ($f'' < 0$).

18.1.7. Failure modes



Three traps with critical points

- **$f'(c) = 0$ does not mean extremum.** For $f(x) = x^3$, $f'(0) = 0$ but the function rises straight through: f' is positive on both sides. This is a **stationary**

inflection point.

- **$f''(c) = 0$ is inconclusive.** x^4, $-x^4$ and x^3 all have $f'(0) = f''(0) = 0$; they have a minimum, a maximum and neither, respectively. Use the first derivative test.
- **Corners are critical points too.** $|x|$ has its minimum at $x = 0$, where f' does not exist. Setting $f'(x) = 0$ alone would miss it.
- **Inflection needs a change.** $f''(c) = 0$ is not enough for an inflection point; the sign of f'' must actually change (x^4 has $f''(0) = 0$ but no inflection).

18.1.8. Where you'd use this

- **Medicine: peak drug level.** After a tablet, the concentration in the blood is often modelled as $C(t) = 20t\,e^{-0.5t}$ mg/L (t in hours). Product rule: $C'(t) = 20e^{-0.5t}(1 - 0.5t)$, which is zero only at $t = 2$; the sign goes $+ \to -$, so the peak is at 2 hours, $C(2) = 40/e \approx 14.7$ mg/L. Pharmacologists call these $t_{\max}$ and $C_{\max}$, and they decide dosing intervals.
- **Everyday: reading an epidemic or sales curve.** On a graph of *total* cases (or total downloads of an app), the inflection point is the moment the daily count peaks: before it the curve is concave up (each day worse than the last), after it concave down (still rising, but more slowly). "We have passed the inflection point" is a claim about the sign of f''.
- **Engineering.** A beam's bending moment is largest where its derivative, the shear force, is zero; structural engineers find the critical section with exactly this first derivative test.
- **Computing.** Peak detection in signals (finding heartbeats in an ECG trace, or onsets in audio) looks for sign changes of the discrete slope $y_{k+1} - y_k$, the first derivative test applied to samples. And optimisers check convexity ($f'' \geq 0$ everywhere) because a convex loss has no traps: every local minimum is global.

US curriculum

Increasing/decreasing intervals and the first and second derivative tests are core **AP Calculus AB** topics ("Analytical Applications of Differentiation"), usually grade 11 or 12. Precalculus classes describe increasing/decreasing intervals from graphs without derivatives.

18.2. Curve sketching as a procedure

You have a function and want an honest picture of it — not a plot from a few sampled points that might miss a wiggle, but a sketch whose every feature is justified. As a programmer you will recognise what follows: it is a pipeline that collects facts and then renders them.

Example 9. Procedure — Sketching the graph of $y = f(x)$

Input: a formula for f.

Steps:

1. **Domain.** Where is f defined? (Watch for division by zero, even roots of negatives, logarithms of non-positives.)
2. **Intercepts.** y-intercept: $f(0)$. x-intercepts: solve $f(x) = 0$ if feasible.
3. **Symmetry.** Even ($f(-x) = f(x)$, mirror in the y-axis) or odd ($f(-x) = -f(x)$, rotational symmetry about the origin)?
4. **Asymptotes and end behaviour.** Limits as $x \to \pm\infty$, and at any excluded points (Chapter 16).
5. **First derivative.** Critical points; intervals of increase/decrease; classify local extrema.
6. **Second derivative.** Intervals of concavity; inflection points.
7. **Tabulate** the key points with their y-values.
8. **Draw:** plot the key points and connect them, obeying the rise/fall and bend information on each interval.

Output: a sketch with every max, min, inflection, intercept and asymptote correctly placed.

18.2.1. Worked example

Sketch $f(x) = x^4 - 4x^3$.

1. **Domain:** all real x (a polynomial).
2. **Intercepts:** $f(0) = 0$. Solve $x^4 - 4x^3 = x^3(x - 4) = 0$: $x = 0$ or $x = 4$.
3. **Symmetry:** $f(-x) = x^4 + 4x^3$, which is neither $f(x)$ nor $-f(x)$. No symmetry.
4. **End behaviour:** the leading term x^4 dominates, so $f(x) \to +\infty$ as $x \to \pm\infty$.
5. **First derivative:**

$$f'(x) = 4x^3 - 12x^2 = 4x^2(x - 3).$$

   Critical points: $x = 0$ and $x = 3$. Signs: $4x^2 \geq 0$ always, so the sign of f' is the sign of $x - 3$ (except at $x = 0$): negative for $x < 3$, positive for $x > 3$. So f decreases on $(-\infty, 3)$ and increases on $(3, \infty)$. At $x = 3$: $- \to +$, a local (in fact global) minimum, $f(3) = 81 - 108 = -27$. At $x = 0$: no sign change — a stationary point that is *not* an extremum.
6. **Second derivative:**

$$f''(x) = 12x^2 - 24x = 12x(x - 2).$$

Zero at $x = 0$ and $x = 2$. Signs: positive for $x < 0$, negative on $(0, 2)$, positive for $x > 2$. Concavity changes at both, so both are inflection points: $(0, 0)$ and $(2, 16 - 32) = (2, -16)$.

7. **Key points:**

x	$f(x)$	feature
0	0	intercept; stationary inflection point (flat, bending changes)
2	-16	inflection point (falling, cap turns to cup)
3	-27	global minimum
4	0	intercept

8. **Draw:** come down from the upper left in a cup shape, flatten to a horizontal tangent at the origin while switching to a cap, keep falling and switch back to a cup at $(2, -16)$, bottom out at $(3, -27)$, rise through $(4, 0)$ and up to the right.

Notice what the procedure caught that random sampling might miss: the origin is flat but not a turning point. Point-sampling plots tend to show it as a slight kink or not at all.

Computer-science connection: why plotting libraries still need you

A plotting library samples f at, say, 500 evenly spaced points and joins them with line segments. That is fast and usually fine, but it can miss narrow spikes, draw spurious vertical lines across asymptotes (joining $+10^6$ to -10^6 across $x = 0$ for $1/x$), and hide flat stationary points. *Adaptive* plotters refine the sampling where the curve bends sharply — that is, where $|f''|$ is large — which is the curve-sketching procedure automated. Knowing where the critical points are tells you which plot window to choose in the first place.

18.3. Optimisation

18.3.1. The problem

"Optimisation" is mathematics for the question every engineer asks: *what is the best choice?* The cheapest design, the shortest route, the highest throughput, the smallest error. In the single-variable form it reads: find the value of x in some allowed set that makes a quantity $f(x)$ as large (or as small) as possible.

18.3.2. Intuition and why it works

A smooth function on a closed interval $[a, b]$ has to achieve both a highest and a lowest value somewhere — this is the **Extreme Value Theorem**. The highest point can only be in one of two kinds of place:

- at an **endpoint** a or b (the function might just keep rising until it is stopped by the boundary), or
- at an **interior** point where the graph turns over — and there, if f is differentiable, the tangent must be horizontal: $f'(c) = 0$.

Why horizontal? If $f'(c) > 0$ at an interior point you could step slightly right and go higher; if $f'(c) < 0$ you could step slightly left and go higher. Either way c would not be the top. Only $f'(c) = 0$ (or a point where f' fails to exist) survives. This is **Fermat's theorem** on stationary points, and it reduces an infinite search over every x to a short list of candidates.



Objective function and constraint

In an optimisation problem, the **objective function** is the quantity you want to maximise or minimise (volume, area, cost, error). A **constraint** is a condition the variables must satisfy (fixed fence length, fixed volume). The **feasible set** or **domain** is the set of values satisfying all constraints. The best value of the objective is the **optimum**; the x achieving it is the **optimiser** ($\arg\max$ or $\arg\min$).

18.3.3. The method

Example 10. Procedure — Solving an optimisation word problem

Input: a situation described in words.

Steps:

1. **Draw a picture** and name the quantities with letters, including units.
2. **Write the objective** — the quantity to optimise — as a formula. It will often involve two or more variables.
3. **Use the constraint** to eliminate all but one variable, giving the objective as $f(x)$.
4. **State the domain** of x from the physical situation (lengths positive, can't cut more than the sheet, …).
5. **Differentiate**, solve $f'(x) = 0$, and list critical points in the domain.
6. **Compare candidates:** evaluate f at each critical point and at the endpoints of the domain (closed-interval method); or, on an open domain, confirm with the first or second derivative test.
7. **Answer the question that was asked** — in words, with units. (Often it asks for the dimensions, not the optimal value, or both.)

Output: the optimal choice and the optimal value.

18.3.4. Worked example 1: the open box

A square sheet of card is 20 cm on each side. You cut an equal square of side x from each corner and fold up the flaps to make an open-topped box. What x gives the largest volume? *Try it:* Design the open box ; Graph the box volume .

Picture and names. After cutting, the base is a square of side $20 - 2x$ (you lose x at each end), and the height is x.

Objective. Volume = base area × height:

$$V(x) = x(20 - 2x)^2.$$

There was only one free variable to begin with, so no constraint is needed to eliminate anything.

Domain. x must be positive, and $20 - 2x$ must be positive, so $0 \le x \le 10$ (allowing the degenerate endpoint boxes, which have zero volume).

Differentiate. Use the product rule with $u = x$, $v = (20 - 2x)^2$, so $u' = 1$ and, by the chain rule, $v' = 2(20 - 2x) \cdot (-2) = -4(20 - 2x)$:

$$\begin{aligned} V'(x) &= 1 \cdot (20 - 2x)^2 + x \cdot \big(-4(20 - 2x)\big) && \text{(product rule)} \\ &= (20 - 2x)\big[(20 - 2x) - 4x\big] && \text{(factor out } 20 - 2x) \\ &= (20 - 2x)(20 - 6x). \end{aligned}$$

Critical points. $V'(x) = 0$ when $20 - 2x = 0$ ($x = 10$, an endpoint) or $20 - 6x = 0$ ($x = 10/3 \approx 3.333$).

Compare candidates.

$$V(0) = 0, \qquad V(10) = 10 \cdot 0^2 = 0,$$
$$V\left(\tfrac{10}{3}\right) = \tfrac{10}{3}\left(20 - \tfrac{20}{3}\right)^2 = \tfrac{10}{3}\left(\tfrac{40}{3}\right)^2 = \tfrac{10}{3} \cdot \tfrac{1600}{9} = \tfrac{16000}{27} \approx 592.6.$$

Answer. Cut squares of side $10/3 \approx 3.33$ cm. The box is $13.33 \times 13.33 \times 3.33$ cm and holds about $593\ \text{cm}^3$.

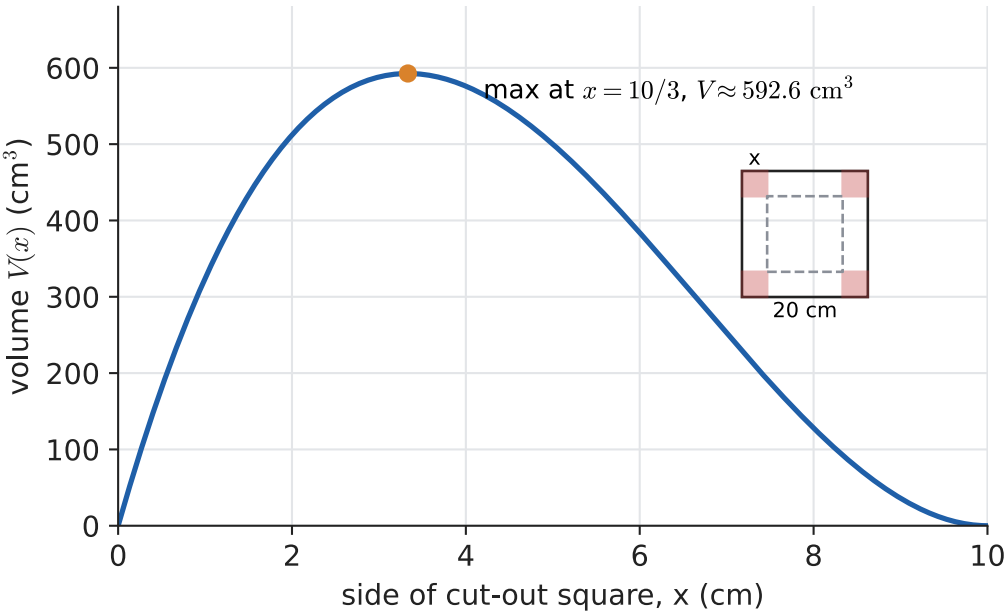


Figure 25. Volume of the box as a function of the cut size. The curve is zero at both ends of the domain and peaks where $V'(x) = 0$.

18.3.5. Worked example 2: fencing along a river

A farmer has 200 m of fencing to enclose a rectangular field along a straight river. No fence is needed along the river. What dimensions maximise the area?

Names. Let x be the length of each of the two sides perpendicular to the river and y the side parallel to it.

Objective: $A = xy$. **Constraint:** fence used $= 2x + y = 200$.

Eliminate. $y = 200 - 2x$, so

$$A(x) = x(200 - 2x) = 200x - 2x^2, \qquad 0 \le x \le 100.$$

Differentiate and solve.

$$A'(x) = 200 - 4x = 0$$
$$x = 50.$$

Check. $A''(x) = -4 < 0$, so concave down everywhere: $x = 50$ is the global maximum. (Endpoints give $A(0) = A(100) = 0$.)

Answer. $x = 50$ m, $y = 200 - 100 = 100$ m, area $5000\ \text{m}^2$. The side along the river is twice each of the other sides.

Alternative approach. $A(x) = 200x - 2x^2$ is a downward parabola; completing the square (Volume 1, Chapter 8) gives $A = -2(x - 50)^2 + 5000$, so the vertex is at $x = 50$ — no calculus needed. For quadratics, the vertex formula is the quicker method. Calculus earns its keep when the objective is not a quadratic, as in the next example.

18.3.6. Worked example 3: minimal cost of a can

A drinks company wants a closed cylindrical can holding $500\ \text{cm}^3$. Metal costs the same per square centimetre everywhere, so minimising cost means minimising surface area. What radius and height should they use? *Try it:* Find the cheapest can .

Names. Radius r, height h, both in cm.

Objective. Surface area = two circular ends + the curved side (which unrolls into a rectangle of width $2\pi r$, height h):

$$S = 2\pi r^2 + 2\pi r h.$$

Constraint. $\pi r^2 h = 500$, so $h = \dfrac{500}{\pi r^2}$.

Eliminate.

$$\begin{aligned} S(r) &= 2\pi r^2 + 2\pi r \cdot \frac{500}{\pi r^2} \\ &= 2\pi r^2 + \frac{1000}{r}, \qquad r > 0. \end{aligned}$$

Differentiate. Write $1000/r = 1000 r^{-1}$:

$$\begin{aligned} S'(r) = 4\pi r - 1000 r^{-2} = 4\pi r - \frac{1000}{r^2} &= 0 \\ 4\pi r &= \frac{1000}{r^2} \\ r^3 &= \frac{1000}{4\pi} = \frac{250}{\pi} \\ r &= \left(\frac{250}{\pi}\right)^{1/3} \approx 4.301\ \text{cm}. \end{aligned}$$

Check. $S''(r) = 4\pi + 2000/r^3 > 0$ for all $r > 0$: concave up on the whole domain, so this single critical point is the global minimum. (The domain $r > 0$ is open — there are no endpoints to check — but as $r \to 0^+$ or $r \to \infty$, $S \to \infty$, confirming a minimum in between.)

Height. Here is the elegant part. From $r^3 = 250/\pi$ we get $\pi r^3 = 250$, so

$$h = \frac{500}{\pi r^2} = \frac{2 \cdot 250}{\pi r^2} = \frac{2\pi r^3}{\pi r^2} = 2r \approx 8.603\ \text{cm}.$$

Answer. $r \approx 4.30$ cm, $h \approx 8.60$ cm, surface area $S \approx 348.7\ \text{cm}^2$. The cheapest can is exactly as tall as it is wide. (Real drinks cans are taller and thinner, because thicker metal is used for the ends, the ends are more expensive to form, and a tall can is easier to hold — change the cost per unit area of the ends in exercise 6 and watch the optimum move.)

18.3.7. Alternative approaches and choosing a method

- **Algebra alone:** for quadratic objectives, the vertex of a parabola (completing the square) is fastest. For products with a fixed sum, the AM–GM inequality ($\sqrt{ab} \le \frac{a+b}{2}$, equality when $a = b$) solves many problems in one line.
- **Calculus:** the general tool for any differentiable objective in one variable.
- **Numerical search:** when $f'(x) = 0$ can't be solved by hand — apply Newton's method (Section 18.10) to f', or evaluate f on a fine grid. For many variables, gradient descent (Section 18.11).

Choose the lightest tool that works. If the objective is a quadratic, don't differentiate it twice and run the second-derivative test; just find the vertex.



Classic optimisation mistakes

- **Forgetting the endpoints.** On a closed interval, the extreme value may be at a boundary where $f' \neq 0$. Example: maximise $f(x) = x^2$ on $[-1, 3]$. The only critical point, $x = 0$, is the *minimum*; the maximum is $f(3) = 9$ at an endpoint.
- **Optimising the wrong thing.** Read the question: "minimise cost" with different prices per material is not "minimise area".
- **Ignoring the domain.** A critical point at $x = -2$ cm is not a box.
- **Local is not global.** A local maximum may not be the global one. Compare all candidates, or show the function is concave down (or up) on the whole domain.

18.3.8. Where you'd use this

- **Everyday and business: setting a price.** A small company sells phone cases that cost £10 each to make. At price p pounds it sells $q = 1000 - 20p$ a month. Profit is $P(p) = (p - 10)(1000 - 20p)$, so $P'(p) = (1000 - 20p) - 20(p - 10) = 1200 - 40p$, zero at $p = 30$. At £30 it sells 400 cases for a profit of $20 \times 400 = 8000$ pounds a month; at £25 or £35 the profit is only £7500. Economists phrase the same calculation as "marginal revenue equals marginal cost".
- **Engineering:** minimum material for a given strength (the can and box above are the toy versions); optimal beam cross-sections.
- **Physics:** light travels the path of least time (Fermat's principle); setting the derivative of travel time to zero gives Snell's law of refraction, the rule behind lenses and fibre optics.
- **Computing: how often to checkpoint.** A long computation saves a checkpoint every τ seconds; each save costs C seconds, and on a crash you lose on average half an interval. With failures every M seconds on average, the time wasted per second is about $W(\tau) = \frac{C}{\tau} + \frac{\tau}{2M}$. Then $W'(\tau) = -\frac{C}{\tau^2} + \frac{1}{2M} = 0$ gives $\tau = \sqrt{2CM}$ (Young's formula). For a 60-second checkpoint and one failure a day ($M = 86\,400$ s), $\tau = \sqrt{2 \cdot 60 \cdot 86\,400} \approx 3220$ s: checkpoint about every 54 minutes. The same "one cost falls, one rises" shape governs cache sizes, batch sizes and retry timeouts.

Computer-science connection: optimisation is what machine learning does

Training a model means choosing parameters to minimise a **loss function** — a measure of how wrong the model's predictions are. With one parameter, it is exactly this section: set the derivative to zero. Real models have millions of parameters, and $f'(x) = 0$ becomes a system of millions of equations that nobody solves directly. Instead, we walk downhill using the derivative — gradient descent (Section 18.11, and properly in Chapter 20). The vocabulary carries over intact: local minima, critical points ("saddle points" in many dimensions), convex objectives with no bad local minima.



The bees knew first

Pappus of Alexandria (4th century) remarked that bees build hexagonal cells because, among shapes that tile the plane, the hexagon encloses the most honey for the least wax. This "honeycomb conjecture" — that the regular hexagonal tiling has the least total perimeter for cells of a given area — was finally proved by Thomas Hales in 1999. Optimisation questions are easy to state and can be very hard to settle.



Box designer

Drag a slider for the cut size x on a 20 cm sheet and watch a 3-D box fold up, with its volume plotted live as a point on the $V(x)$ curve. A second slider changes the sheet to a rectangle ($a \times b$); see the optimum move, and compare it with the root of $V'(x) = 0$ found by the quadratic formula.

Try it: Open the box designer · Try a rectangular sheet · Box volume V(x) with sliders

US curriculum

Optimisation word problems are a signature topic of **AP Calculus AB** (and of the first semester of college calculus). Vertex-of-parabola optimisation appears earlier, in **Algebra I/II**.

18.4. Exercises: shape and optimisation

1. **(Recognition)** At a point c, $f'(c) = 0$ and $f''(c) = 5$. What can you conclude? What if $f''(c) = 0$?
2. **(Application)** Find and classify the critical points of $g(x) = x^3 - 12x$. Where is g concave up?
3. **(Application)** Find the global maximum and minimum of $h(x) = x^3 - 3x$ on the interval $[0, 3]$.
4. **(Application)** Two positive numbers add to 20. Which pair has the largest product?
5. **(Problem solving)** A rectangle has perimeter 400 m (fenced on all four sides). Show that the area is maximised by a square.

6. **(Problem solving)** Redo the can problem with the ends costing twice as much per $\mathbf{cm^2}$ as the side. Show that the optimal can now has $h = 4r$. *Try it:* Ends cost double: h = 4r .

▼ **Answers**

1. $f''(c) > 0$ means concave up at a stationary point: a local minimum. If $f''(c) = 0$ the second derivative test is inconclusive; use the first derivative test (e.g. x^4, $-x^4$, x^3 at 0 give min, max, neither).
2. $g'(x) = 3x^2 - 12 = 3(x-2)(x+2)$, so critical points $x = \pm 2$. $g''(x) = 6x$: $g''(-2) = -12 < 0$, so local max $g(-2) = -8 + 24 = 16$; $g''(2) = 12 > 0$, so local min $g(2) = 8 - 24 = -16$. Concave up where $6x > 0$, i.e. $x > 0$; inflection at $(0, 0)$.
3. $h'(x) = 3x^2 - 3 = 3(x-1)(x+1)$; in $[0, 3]$ the only critical point is $x = 1$. Candidates: $h(0) = 0$, $h(1) = 1 - 3 = -2$, $h(3) = 27 - 9 = 18$. Global max 18 at $x = 3$ (an endpoint), global min -2 at $x = 1$.
4. Numbers x and $20 - x$; $P(x) = 20x - x^2$, $P'(x) = 20 - 2x = 0$ gives $x = 10$; $P'' = -2 < 0$. Both numbers are 10, product 100.
5. Sides x and y with $2x + 2y = 400$, so $y = 200 - x$ and $A(x) = x(200 - x) = 200x - x^2$. $A'(x) = 200 - 2x = 0 \Rightarrow x = 100$, so $y = 100$: a square, area $10\,000\ \mathrm{m}^2$. $A'' = -2 < 0$ confirms a maximum.
6. Cost $C = 2 \cdot (2\pi r^2) + 2\pi r h$ (in units of the side price). With $h = 500/(\pi r^2)$: $C(r) = 4\pi r^2 + 1000/r$, $C'(r) = 8\pi r - 1000/r^2 = 0 \Rightarrow r^3 = 125/\pi$, i.e. $\pi r^3 = 125$. Then $h = 500/(\pi r^2) = 4 \cdot 125/(\pi r^2) = 4\pi r^3/(\pi r^2) = 4r$. $r \approx 3.41$ cm, $h \approx 13.66$ cm — taller and thinner, like a real can.

18.5. Related rates

18.5.1. The problem

Air is pumped into a spherical balloon at a steady $100\ \mathrm{cm}^3$ per second. How fast is the radius growing when the radius is 10 cm? You know one rate (volume per second) and want another (radius per second). The two quantities are tied together by a formula, $V = \frac{4}{3}\pi r^3$, and that link forces their rates to be tied together too.

18.5.2. Intuition and why it works

When several quantities all change with time, each is a function of t. If an equation relates them at every instant, you can differentiate *both sides of the equation with respect to* t — the two sides are equal functions of time, so their rates of change are equal. The chain rule (Chapter 17) does the work: the derivative of r^3 with respect to time is $3r^2 \cdot \frac{dr}{dt}$, "the rate V changes per unit of r, times the rate r changes per unit of time". This is implicit differentiation with time as the hidden variable.

Example 11. Procedure — Related rates

Input: a situation with several changing quantities, one known rate, one wanted rate, and the "snapshot" values at the instant of interest.

Steps:

1. Draw a picture; name every changing quantity as a function of time.
2. Write down the known rate and the wanted rate as derivatives (e.g. $\frac{dV}{dt} = 100$, want $\frac{dr}{dt}$).
3. Find an equation linking the quantities that holds *at all times*.
4. Differentiate both sides with respect to t (chain rule on every changing quantity).
5. Only now substitute the snapshot values, and solve for the wanted rate.
6. State the answer with units and sign (negative means decreasing).

Output: the wanted rate at the given instant.

18.5.3. Worked example 1: the balloon

Known: $\frac{dV}{dt} = 100\ \text{cm}^3/\text{s}$. **Wanted:** $\frac{dr}{dt}$ when $r = 10$ cm. **Link:** $V = \frac{4}{3}\pi r^3$.

$$\frac{dV}{dt} = \tfrac{4}{3}\pi \cdot 3r^2 \frac{dr}{dt} = 4\pi r^2 \frac{dr}{dt} \qquad \text{(differentiate w.r.t. } t\text{; chain rule)}$$

$$100 = 4\pi(10)^2 \frac{dr}{dt} \qquad \text{(now substitute the snapshot)}$$

$$\frac{dr}{dt} = \frac{100}{400\pi} = \frac{1}{4\pi} \approx 0.0796\ \text{cm/s}.$$

Notice that $4\pi r^2$ is the balloon's surface area: the new air is spread as a thin shell over the whole surface, so the bigger the balloon, the slower the radius grows.

18.5.4. Worked example 2: the sliding ladder (a Mechanics classic)

A 5 m ladder leans against a vertical wall. Its foot slides away from the wall at 0.5 m/s. How fast is the top sliding down the wall when the foot is 3 m from the wall?

Names. $x(t)$: distance of the foot from the wall; $y(t)$: height of the top. **Known:** $\frac{dx}{dt} = 0.5$. **Wanted:** $\frac{dy}{dt}$ when $x = 3$. **Link** (Pythagoras, true at all times): $x^2 + y^2 = 25$.

Snapshot: when $x = 3$, $y = \sqrt{25 - 9} = 4$.

$$
\begin{aligned}
2x\frac{dx}{dt} + 2y\frac{dy}{dt} &= 0 && \text{(differentiate; 25 is constant)}\\
x\frac{dx}{dt} + y\frac{dy}{dt} &= 0 && \text{(divide by 2)}\\
3(0.5) + 4\frac{dy}{dt} &= 0 && \text{(substitute } x=3, y=4, \tfrac{dx}{dt}=0.5\text{)}\\
\frac{dy}{dt} &= -\frac{1.5}{4} = -0.375 \text{ m/s}.
\end{aligned}
$$

The top slides *down* (negative sign) at 0.375 m/s. As the top approaches the ground ($y \to 0$), $\frac{dy}{dt} = -\frac{x}{y}\frac{dx}{dt}$ grows without bound — the model breaks down because in reality the ladder's top leaves the wall.



Substituting too early

Never plug in the snapshot values before differentiating. If you wrote $x = 3$ into $x^2 + y^2 = 25$ first, you would get $9 + y^2 = 25$, and differentiating that says $2y\frac{dy}{dt} = 0$ — nonsense, because you have frozen x at one instant and thrown its rate away. Differentiate the general relationship, *then* substitute.

18.5.5. Where you'd use this

- **Navigation and radar.** A radar dish measures how fast the *distance* to an aircraft changes, not the aircraft's speed. A plane at 10 km altitude flies level at 800 km/h straight towards a point above the radar. With horizontal distance x and slant distance s, $s^2 = x^2 + 10^2$, so $s\frac{ds}{dt} = x\frac{dx}{dt}$. When $x = 24$ km, $s = 26$ km, and the radar sees $\frac{ds}{dt} = \frac{24}{26} \times (-800) \approx -738$ km/h: the closing speed is less than the true speed, and the related-rates equation is how the system corrects for it. Police speed guns aimed from the roadside have the same "cosine error", which always *underestimates* your speed.
- **Medicine.** Doctors track a tumour by its diameter on a scan, but treatment depends on volume. For a sphere of diameter D, $V = \frac{\pi}{6}D^3$, so $\frac{dV}{dt} = \frac{\pi}{2}D^2\frac{dD}{dt}$. A 2 cm tumour growing 1 mm a month is adding $\frac{\pi}{2}(4)(0.1) \approx 0.63$ cm³ a month, and the same diametric growth at 4 cm means four times the volume growth.
- **Computing.** In games and animation, a camera tracking a moving target must turn at the rate $\frac{d\theta}{dt}$ implied by the target's speed and distance (differentiate $\tan\theta = y/x$); a target passing close by makes the required turn rate spike, which is why fast fly-bys look jerky without smoothing.

US curriculum

Related rates are a standard **AP Calculus AB** unit ("Contextual Applications of Differentiation"), always featuring ladders, balloons, conical tanks and shadows.

18.6. Linear approximation and differentials

18.6.1. The problem

Without a calculator, what is $\sqrt{4.1}$? You know $\sqrt{4} = 2$ exactly, and 4.1 is very close to 4. Surely $\sqrt{4.1}$ is just a little more than 2 — but how much more?

18.6.2. Intuition

Zoom in on any smooth curve far enough and it looks straight — it looks like its tangent line. So near a point a where you know everything, replace the curve by its tangent line and evaluate the line instead. Lines are trivial to evaluate.

18.6.3. Why it works

The tangent line at $x = a$ passes through $(a, f(a))$ with slope $f'(a)$. Point-slope form (Volume 1, Chapter 5) gives

$$L(x) = f(a) + f'(a)\,(x - a).$$

This L is the **linearisation** of f at a. Because $f'(a)$ is the limit of the slopes $\frac{f(x)-f(a)}{x-a}$, the gap $f(x) - L(x)$ shrinks *faster than* $x - a$ itself as $x \to a$. For twice-differentiable functions the error is roughly $\frac{1}{2}f''(a)(x-a)^2$: halve the distance, quarter the error. That quadratic error term will reappear in Taylor polynomials and as the reason Newton's method converges so quickly.



Linearisation, differential

The **linearisation** (or **tangent line approximation**) of f at a is $L(x) = f(a) + f'(a)(x - a)$, and $f(x) \approx L(x)$ for x near a.

The **differentials** dx and dy restate this: dx is a small change in the input, and $dy = f'(x)\,dx$ is the corresponding change *along the tangent line*. The true change is $\Delta y = f(x + dx) - f(x)$, and $\Delta y \approx dy$. This is where Leibniz's notation $\frac{dy}{dx}$ earns its look: $dy = \frac{dy}{dx}\,dx$.

18.6.4. Worked example: $\sqrt{4.1}$

Take $f(x) = \sqrt{x} = x^{1/2}$ and $a = 4$. *Try it:* Estimate √4.1 with a tangent .

$$\begin{aligned} f(4) &= 2 \\ f'(x) &= \tfrac{1}{2}x^{-1/2} = \frac{1}{2\sqrt{x}}, \qquad f'(4) = \frac{1}{4} \\ L(x) &= 2 + \tfrac{1}{4}(x - 4) \\ \sqrt{4.1} \approx L(4.1) &= 2 + \tfrac{1}{4}(0.1) = 2.025. \end{aligned}$$

The true value is $2.0248457\ldots$, so the error is about 1.5×10^{-4}. Check against the error estimate: $f''(x) = -\frac{1}{4}x^{-3/2}$, so $f''(4) = -\frac{1}{32}$, and $\frac{1}{2} \cdot \frac{1}{32} \cdot (0.1)^2 \approx 1.56 \times 10^{-4}$. The estimate also predicts the sign: since $\sqrt{x}$ is concave down, the tangent lies *above* the curve and the approximation overshoots.

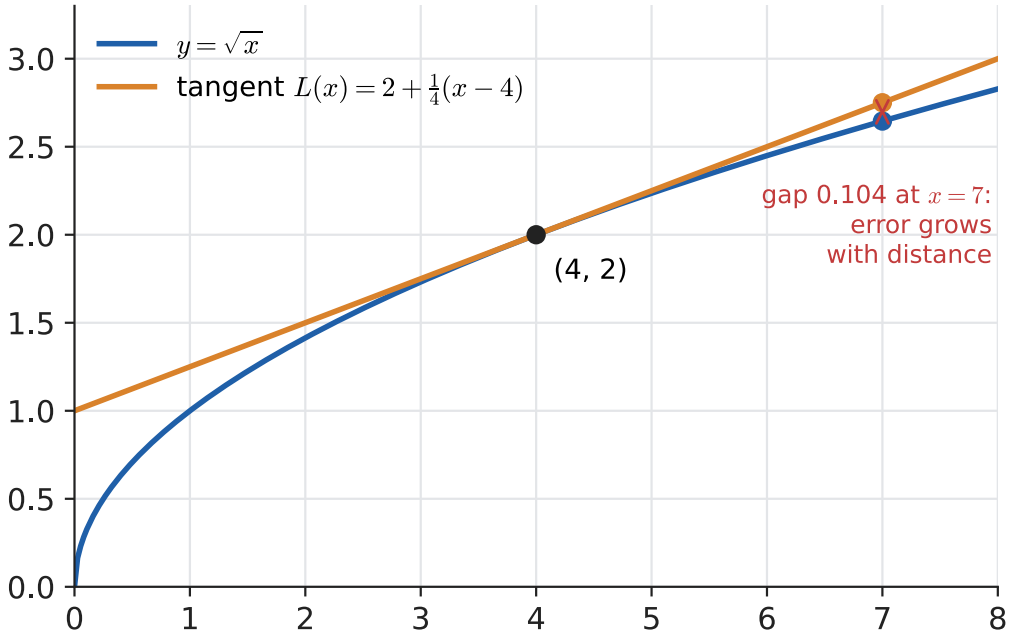


Figure 26. The tangent line at $x = 4$ hugs $\sqrt{x}$ nearby and drifts away further out. Because the curve is concave down, the tangent always overestimates.

18.6.5. Worked example: small angles and the pendulum

Linearise $\sin x$ at $a = 0$: $f(0) = 0$, $f'(0) = \cos 0 = 1$, so $L(x) = x$. For small angles in radians, $\sin x \approx x$. At $x = 0.1$ (about $5.7°$), $\sin 0.1 = 0.0998334$, an error under 0.2%. *Try it:* sin x ≈ x, visually .

This is the step every Mechanics course takes with the simple pendulum. The exact equation of motion contains $\sin\theta$; replacing it with θ turns it into the equation of simple harmonic motion, whose solutions are sines and cosines with period $2\pi\sqrt{\ell/g}$. That famous formula is only true for small swings — it is a linear approximation.

18.6.6. Worked example: error propagation with differentials

You measure a ball's radius as 10 cm, with a possible error of ± 0.1 cm. How uncertain is the computed volume?

$$V = \tfrac{4}{3}\pi r^3, \qquad dV = 4\pi r^2\,dr \qquad \text{(differential)}$$
$$dV = 4\pi(10)^2(0.1) = 40\pi \approx 125.7 \text{ cm}^3.$$

Relative to $V = \frac{4}{3}\pi(1000) \approx 4189 \text{ cm}^3$, that is

$$\frac{dV}{V} = \frac{4\pi r^2\,dr}{\frac{4}{3}\pi r^3} = 3\,\frac{dr}{r} = 3 \times 1\% = 3\%.$$

A 1% error in the radius becomes a 3% error in the volume: cubing triples relative errors. (In general, for $y = x^n$, relative error multiplies by n.)

18.6.7. Where you'd use this

- **Finance: the rule of 70.** How long does money take to double at 7% a year? Exactly, $t = \ln 2 / \ln 1.07 \approx 10.24$ years. The linearisation $\ln(1+r) \approx r$ gives $t \approx 0.693/r$, which bankers round to "70 divided by the percentage": $70/7 = 10$ years. The same shortcut tells you that 3% inflation halves the value of money in about 23 years.
- **Science and engineering.** Error propagation (above) is how lab reports and engineering tolerances work; the small-angle approximation $\sin\theta \approx \theta$ is used by surveyors and astronomers (an object 1 m wide seen at 1 km subtends about 0.001 rad), and by every pendulum clock.
- **Computing.** Graphics and physics engines linearise constantly: a skinned character's joints, a car's suspension and a robot arm are simulated by replacing the nonlinear equations with their tangent-line versions for one small time step at a time. The extended Kalman filter in the sidebar below is the same move for navigation.

Computer-science connection: condition numbers and floating point

The ratio "relative change out / relative change in", $\left|\frac{x f'(x)}{f(x)}\right|$, is the **condition number** of evaluating f at x. For $V = \frac{4}{3}\pi r^3$ it is 3. Because every floating-point number carries a relative rounding error of about 10^{-16} (Chapter 32), the condition number tells you how many of those 16 digits survive a computation. Evaluating $f(x) = x - 1$ near $x = 1$ has condition number $|x/(x-1)|$, which blows up: that is **catastrophic cancellation**, explained by a derivative. Linearisation is also the core of the extended Kalman filter, which tracks robots and aircraft by repeatedly approximating nonlinear motion with tangent lines (Jacobians in many dimensions).

US curriculum

Linear approximation (often called "local linearity") is in **AP Calculus AB**; differentials appear in many college Calculus I courses. Error propagation reappears in high-school and college lab science.

18.7. L'Hôpital's rule

18.7.1. The problem

In Chapter 16 you met limits like $\lim_{x\to 0}\frac{\sin x}{x}$, where plugging in gives $\frac{0}{0}$ — an **indeterminate form**, meaning the answer depends on *how fast* numerator and denominator approach zero. Algebraic tricks and squeezing work sometimes. Is there a general tool? Yes: compare their rates of approach — their derivatives.

18.7.2. Why it works

Suppose $f(a) = g(a) = 0$ and both are differentiable at a with $g'(a) \neq 0$. Near a, replace each by its linearisation:

$$\frac{f(x)}{g(x)} \approx \frac{f(a)+f'(a)(x-a)}{g(a)+g'(a)(x-a)} = \frac{f'(a)(x-a)}{g'(a)(x-a)} = \frac{f'(a)}{g'(a)}.$$

Both functions start at zero; their ratio is governed by their speeds. The full theorem (proved with a generalisation of the Mean Value Theorem) states:

$$\text{If } \lim_{x\to a}\frac{f(x)}{g(x)} \text{ is of the form } \frac{0}{0} \text{ or } \frac{\pm\infty}{\pm\infty}, \text{ then } \lim_{x\to a}\frac{f(x)}{g(x)} = \lim_{x\to a}\frac{f'(x)}{g'(x)}$$

provided the right-hand limit exists (or is $\pm\infty$). It also works for $a = \pm\infty$ and one-sided limits.



Indeterminate form

An **indeterminate form** is a limit shape whose value cannot be decided from the limits of the pieces alone: $\frac{0}{0}$, $\frac{\infty}{\infty}$, $0\cdot\infty$, $\infty-\infty$, 0^0, 1^∞, ∞^0. L'Hôpital applies directly only to the first two; the others are first rewritten as a quotient.

18.7.3. Worked examples

1. $\lim_{x\to 0}\dfrac{\sin x}{x}$: form $\frac{0}{0}$.

$$\lim_{x\to 0}\frac{\sin x}{x} = \lim_{x\to 0}\frac{\cos x}{1} = \cos 0 = 1.$$

(Honesty note: the derivative of $\sin x$ was itself *derived* using this limit in Chapter 17, so this is a consistency check, not an independent proof.)

2. $\lim_{x\to 0}\dfrac{e^x - 1 - x}{x^2}$: form $\frac{0}{0}$. Apply twice:

$$\begin{aligned}\lim_{x\to 0}\frac{e^x-1-x}{x^2} &= \lim_{x\to 0}\frac{e^x-1}{2x} && \text{(still } \tfrac{0}{0}\text{, apply again)}\\ &= \lim_{x\to 0}\frac{e^x}{2} = \frac{1}{2}.\end{aligned}$$

3. $\lim_{x\to\infty}\dfrac{\ln x}{x}$: form $\frac{\infty}{\infty}$.

$$\lim_{x\to\infty}\frac{\ln x}{x} = \lim_{x\to\infty}\frac{1/x}{1} = 0.$$

Logarithms grow more slowly than any line — the calculus behind "$O(\log n)$ beats $O(n)$".

4. $\lim_{x\to 0^+} x\ln x$: form $0\cdot(-\infty)$. Rewrite as a quotient first:

$$\begin{aligned}\lim_{x\to 0^+} x\ln x &= \lim_{x\to 0^+}\frac{\ln x}{1/x} && \text{(now } \tfrac{-\infty}{\infty}\text{)}\\ &= \lim_{x\to 0^+}\frac{1/x}{-1/x^2} = \lim_{x\to 0^+}(-x) = 0.\end{aligned}$$

This limit is why information theory can define $0 \log 0 = 0$ in the entropy formula.



When L'Hôpital does not apply

- **Check the form first.** $\lim_{x \to 0} \frac{x+1}{x+2} = \frac{1}{2}$ by direct substitution. Blindly differentiating gives $\frac{1}{1} = 1$ — wrong, because the form was never $\frac{0}{0}$.
- **It is not the quotient rule.** You differentiate numerator and denominator *separately*, not the fraction.
- **It can go in circles.** $\lim_{x \to \infty} \frac{x}{\sqrt{x^2+1}}$ flips between two forms forever; divide through by x instead (answer: 1).
- **If $\lim f'/g'$ does not exist, the rule tells you nothing** — the original limit may still exist.



A theorem bought and paid for

The rule appeared in the first calculus textbook, *Analyse des infiniment petits* (1696), by the Marquis de l'Hôpital. But the mathematics was Johann Bernoulli's: l'Hôpital had paid Bernoulli a salary for his discoveries, under an agreement that he could use them as he wished. Bernoulli's letters, found in the 20th century, settled the matter.

18.8. Taylor polynomials

18.8.1. The problem

A tangent line matches a function's value and slope at one point. It is good nearby, and then drifts off. Could we do better by matching the *bending* too — the second derivative — and the third, and so on? And how do computers actually evaluate $\sin x$, when the only operations a chip performs are addition and multiplication?

18.8.2. Intuition and why it works

Polynomials are the functions computers can evaluate with just $+$ and $\times$. So look for a polynomial

$$p(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots + c_n x^n$$

that agrees with f at $x = 0$ in value *and in its first n derivatives*. Differentiate p repeatedly and set $x = 0$; each time, every term vanishes except one:

$$\begin{aligned}
p(0) &= c_0 && \Rightarrow c_0 = f(0) \\
p'(0) &= c_1 && \Rightarrow c_1 = f'(0) \\
p''(0) &= 2c_2 && \Rightarrow c_2 = \frac{f''(0)}{2} \\
p'''(0) &= 3 \cdot 2\, c_3 && \Rightarrow c_3 = \frac{f'''(0)}{3!} \\
p^{(k)}(0) &= k!\, c_k && \Rightarrow c_k = \frac{f^{(k)}(0)}{k!}.
\end{aligned}$$

The $k!$ (**k factorial**, $k! = 1 \cdot 2 \cdots k$, from Volume 3, Chapter 24 and Volume 1, Chapter 14) appears because differentiating x^k k times brings down $k, k-1, \ldots, 1$. The result is the **Taylor polynomial** of degree n about 0 (also called a **Maclaurin polynomial**):

$$T_n(x) = f(0) + f'(0)\,x + \frac{f''(0)}{2!}x^2 + \cdots + \frac{f^{(n)}(0)}{n!}x^n = \sum_{k=0}^{n} \frac{f^{(k)}(0)}{k!}x^k.$$

About a general centre a, replace x by $(x-a)$ and evaluate derivatives at a. Degree 1 is exactly the linearisation $L(x)$.



Taylor polynomial

The **degree-n Taylor polynomial** of f about a is $T_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k$, the unique polynomial of degree at most n matching f and its first n derivatives at a . Notation: $f^{(k)}$ is the k-th derivative (brackets distinguish it from a power), with $f^{(0)} = f$. Letting $n \to \infty$ gives the **Taylor series**.

18.8.3. Worked example: $\sin x \approx x - \frac{x^3}{6}$

The derivatives of $\sin x$ cycle with period 4: *Try it:* Add Taylor terms to sin x ; Taylor series of sin x .

k	$f^{(k)}(x)$	$f^{(k)}(0)$
0	$\sin x$	0
1	$\cos x$	1
2	$-\sin x$	0
3	$-\cos x$	-1
4	$\sin x$	0
5	$\cos x$	1

Substituting into the formula, the even terms vanish:

$$\sin x \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots = x - \frac{x^3}{6} + \frac{x^5}{120} - \cdots$$

At $x = 0.5$ (where $\sin 0.5 = 0.4794255\ldots$):

Approximation	Value	Error
x	0.5	2.1×10^{-2}
$x - x^3/6$	$0.5 - 0.0208333 = 0.4791667$	2.6×10^{-4}
$x - x^3/6 + x^5/120$	$0.4791667 + 0.0002604 = 0.4794271$	1.5×10^{-6}

Each extra term gains about two decimal digits here. The same procedure gives the other two essential expansions:

$$\cos x \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots, \qquad e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

(for e^x, every derivative is e^x, which is 1 at 0, so every coefficient is $1/k!$).

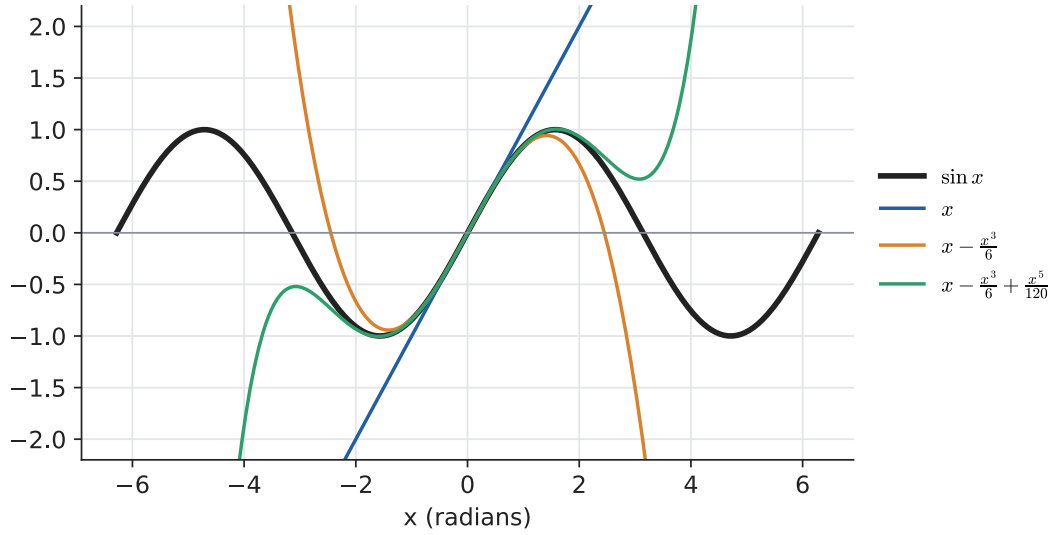


Figure 27. Taylor polynomials of $\sin x$ about 0. Each added term extends the region of good agreement further from the centre.

A short program shows the pattern — accuracy is excellent near the centre and poor far away:

```python
import math

def sin_taylor(x, terms):
    """Sum the first `terms` nonzero terms of x - x^3/3! + x^5/5! - ..."""
    total, term = 0.0, x
    for k in range(terms):
        total += term
        term *= -x * x / ((2 * k + 2) * (2 * k + 3))
    return total

for x in (0.1, 0.5, 1.0, 3.0):
    row = [abs(sin_taylor(x, t) - math.sin(x)) for t in (1, 2, 3, 5)]
    print(f"x = {x:3}: " + "  ".join(f"{e:.1e}" for e in row))
```

```
x = 0.1: 1.7e-04  8.3e-08  2.0e-11  1.4e-17
x = 0.5: 2.1e-02  2.6e-04  1.5e-06  1.2e-11
x = 1.0: 1.6e-01  8.1e-03  2.0e-04  2.5e-08
x = 3.0: 2.9e+00  1.6e+00  3.8e-01  4.2e-03
```

Each new term is computed from the previous one (multiply by $-x^2/((2k+2)(2k+3))$), never from scratch — the factorials are never formed explicitly.

18.8.4. Where you'd use this

- **Physics: where $\frac{1}{2}mv^2$ comes from.** Einstein's kinetic energy is $(\gamma-1)mc^2$ with $\gamma=(1-v^2/c^2)^{-1/2}$. The degree-1 Taylor polynomial of $(1-u)^{-1/2}$ is $1+\frac{1}{2}u$, so $\gamma-1\approx\frac{v^2}{2c^2}$ and the kinetic energy is approximately $\frac{1}{2}mv^2$, the Mechanics formula. At a tenth of the speed of light the two disagree by under 1% ($\gamma-1=0.005038$ against 0.005). *Try it:* Linearise γ: ½mv² .
- **Finance: bond risk.** A bond's price change for a small change Δy in interest rates is written as a Taylor polynomial: $\frac{\Delta P}{P}\approx -D\,\Delta y+\frac{1}{2}C(\Delta y)^2$. The first-order coefficient D is the bond's **duration** and the second-order one its **convexity**. A bond with duration 7 loses about 7% of its value when rates rise by one percentage point.
- **Computing.** Beyond computing $\sin$ (see the sidebar), second-order Taylor expansions are how optimisers such as Newton's method model a loss function locally, and how game physics integrators are analysed for accuracy.

Computer-science connection: how $\sin$ is really computed

A maths library does not sum a long Taylor series. It first uses **range reduction** — the periodicity and symmetries of $\sin$ — to shift any input into a small interval such as $[-\pi/4,\pi/4]$, where a short polynomial is very accurate. The polynomial coefficients are then tuned slightly away from the exact Taylor values (a "minimax" fit) so the worst-case error over the interval is as small as possible, and evaluated with Horner's rule (Volume 1, Chapter 7). Taylor polynomials are the starting idea; numerical analysis (Chapter 32) polishes it.



Know where you are

A Taylor polynomial is a *local* approximation. The table shows five terms of the sine series still miss by 4×10^{-3} at $x=3$. And some functions' series only work in a limited range: the series of $\ln(1+x)$ converges only for $-1<x\le1$, however many terms you take. Always use radians — the derivative of $\sin x$ is $\cos x$ only in radians.

Try it: Where ln(1 + x) stops converging .

US curriculum

Taylor polynomials and series, with error bounds, are part of **AP Calculus BC** (not AB), typically grade 12. L'Hôpital's rule is in both AB and BC.

18.9. Exercises: rates, approximation, limits

1. **(Recognition)** Write down the linearisation of $f(x) = e^x$ at $a = 0$.
2. **(Application)** A stone dropped in a pond makes a circular ripple whose radius grows at 2 cm/s. How fast is the area growing when the radius is 10 cm?
3. **(Application)** Use a linear approximation to estimate $\sqrt[3]{8.3}$ and $\ln(1.05)$. Are your estimates too high or too low? *Try it:* Estimate ∛8.3 with a tangent .
4. **(Application)** Evaluate $\lim_{x \to 0} \dfrac{1 - \cos x}{x^2}$ and $\lim_{x \to \infty} \dfrac{x^2}{e^x}$.
5. **(Problem solving)** Use the degree-2 Taylor polynomial of e^x to estimate $e^{0.1}$, and explain why $1 - \cos x \approx x^2/2$ for small x agrees with your answer to question 4. *Try it:* e^0.1 from a Taylor polynomial .

▼ **Answers**

1. $f(0) = 1$, $f'(0) = e^0 = 1$, so $L(x) = 1 + x$.
2. $A = \pi r^2$, so $\frac{dA}{dt} = 2\pi r \frac{dr}{dt} = 2\pi(10)(2) = 40\pi \approx 125.7 \text{ cm}^2/\text{s}$.
3. $f(x) = x^{1/3}$, $a = 8$: $f(8) = 2$, $f'(x) = \frac{1}{3}x^{-2/3}$, $f'(8) = \frac{1}{3 \cdot 4} = \frac{1}{12}$. $\sqrt[3]{8.3} \approx 2 + 0.3/12 = 2.025$ (true: 2.02469). $\ln$ at $a = 1$: $\ln 1 = 0$, derivative $1/x = 1$, so $\ln 1.05 \approx 0.05$ (true: 0.04879). Both too high: both functions are concave down, so tangent lines lie above them.
4. First: form $\frac{0}{0}$; L'Hôpital gives $\lim \frac{\sin x}{2x} = \frac{1}{2} \lim \frac{\sin x}{x} = \frac{1}{2}$. Second: form $\frac{\infty}{\infty}$; twice gives $\lim \frac{2x}{e^x} = \lim \frac{2}{e^x} = 0$.
5. $e^{0.1} \approx 1 + 0.1 + 0.01/2 = 1.105$ (true: 1.10517). From $\cos x \approx 1 - x^2/2$, $1 - \cos x \approx x^2/2$, so $\frac{1 - \cos x}{x^2} \approx \frac{1}{2}$ for small x, matching the limit.

18.10. Newton's method

This is the chapter's centrepiece, and one of the most important algorithms in all of applied mathematics. We will build it in ten steps — what problem it solves, why that problem is hard, why a tangent line helps, how the formula falls out of the tangent, how to start, how to step, how to repeat, when to stop, when it works brilliantly, and when it fails — and only then hand it to Python.

18.10.1. 1. The problem: solve $f(x) = 0$

Many questions reduce to finding a **root** (or **zero**) of a function: a number r with $f(r) = 0$.

- What is $\sqrt{2}$? It is the positive root of $f(x) = x^2 - 2$.

- Where does $\cos x$ equal x? At the root of $f(x) = \cos x - x$.
- Where is g maximised? At a root of $f = g'$ (Section 18.3).
- What interest rate makes a loan's payments add up to its price? The root of a polynomial of high degree.

Any equation **left** $=$ **right** can be rewritten as **left** $-$ **right** $= 0$, so "find a root" is the universal form of "solve an equation".



Root, zero, iteration

A **root** or **zero** of f is a value r with $f(r) = 0$; graphically, where the curve crosses or touches the x-axis. An **iterative method** produces a sequence of approximations $x_0, x_1, x_2, \ldots$ (read "x-nought, x-one, ..."; the subscript is the step number), each computed from the previous one by the same rule. x_0 is the **initial guess**. The method **converges** if $x_n \to r$ as $n \to \infty$ (a limit of a sequence, Volume 1, Chapter 14).

18.10.2. 2. Why solving directly is often impossible

For linear and quadratic equations we have formulas (Volume 1, Chapter 5 , Volume 1, Chapter 8). Cubics and quartics have formulas too, but they are monstrous. For polynomials of degree five and higher, the Abel–Ruffini theorem says there is *no* general formula in radicals at all. And equations that mix kinds of function, like $\cos x = x$ or $e^x = 3x$, typically have no closed-form solution in terms of familiar functions.

Yet these equations have perfectly definite answers, and we need them to 15 digits. The way forward is to stop looking for a formula for the answer, and instead look for a *procedure that improves a guess*.

18.10.3. 3. Why a tangent line helps

We cannot easily solve $f(x) = 0$ for a curved f. But we can *trivially* solve $L(x) = 0$ for a straight line L. And from Section 18.6 we know that near any point, a smooth curve is well approximated by its tangent line.

So: stand at your current guess x_n on the curve, at the point $(x_n, f(x_n))$. Draw the tangent there. Follow it down (or up) to where it crosses the x-axis. Because the tangent hugs the curve, the place where the *tangent* hits zero should be close to where the *curve* hits zero — closer than where you started. Take that crossing as the new guess and repeat.

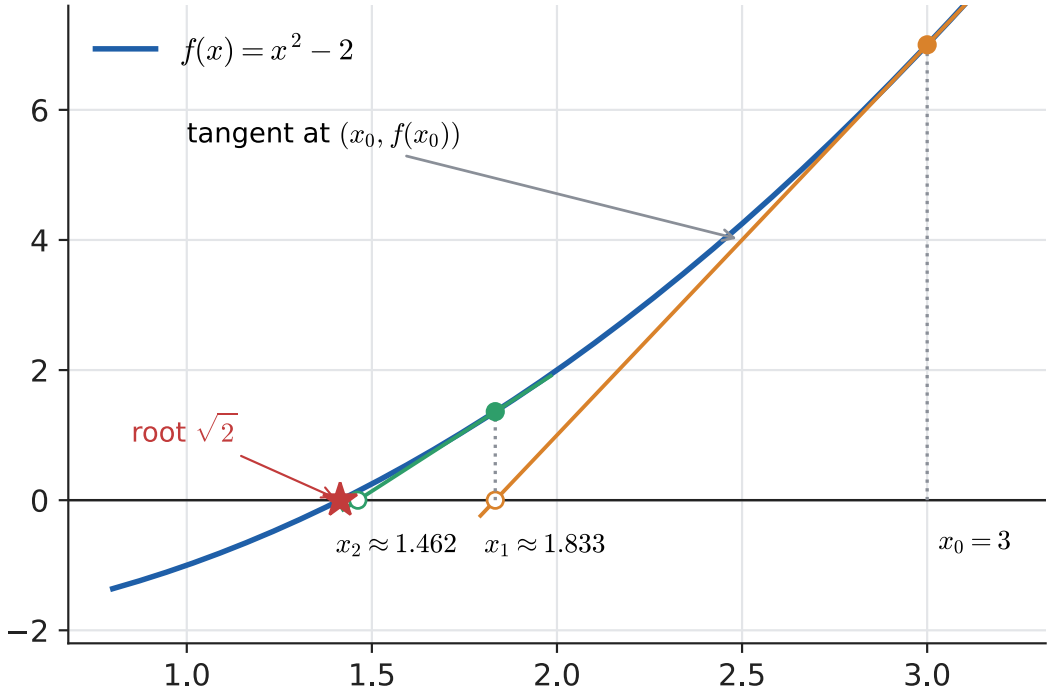


Figure 28. One picture of Newton's method. Two tangent steps take the guess from $x_0 = 3$ to $x_1 \approx 1.833$ to $x_2 \approx 1.462$, closing in on $\sqrt{2} \approx 1.414$.

18.10.4. 4. Deriving the formula from the tangent

The tangent at x_n has slope $f'(x_n)$ and passes through $(x_n, f(x_n))$. In point-slope form:

$$y = f(x_n) + f'(x_n)\,(x - x_n).$$

Call its x-intercept x_{n+1}. At that point $y = 0$:

$$\begin{aligned}
0 &= f(x_n) + f'(x_n)\,(x_{n+1} - x_n) && \text{(set } y = 0 \text{ at } x = x_{n+1}\text{)} \\
-f(x_n) &= f'(x_n)\,(x_{n+1} - x_n) && \text{(subtract } f(x_n)\text{)} \\
-\frac{f(x_n)}{f'(x_n)} &= x_{n+1} - x_n && \text{(divide by } f'(x_n)\text{, assuming it is not 0)} \\
x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} && \text{(add } x_n\text{)}
\end{aligned}$$

That is **Newton's method** (also called the **Newton–Raphson method**):

$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}$$

Read it in words: *the correction is the current height divided by the current slope.* If you are at height $f(x_n) = 6$ on a slope of 6, a straight line would reach zero after stepping back 1 unit. The minus sign points the step in the right direction automatically: positive height with positive slope means step left, and so on. Notice also the division by $f'(x_n)$ in the third line — we have already spotted the method's Achilles' heel.

18.10.5. 5. Choosing an initial value

Newton's method needs a starting guess x_0, and the quality of that guess matters more than for any other step. Good ways to get one:

- **Sketch or plot f** and read off roughly where it crosses the axis.
- **Find a sign change.** If $f(a)$ and $f(b)$ have opposite signs and f is continuous, there is a root between a and b (the Intermediate Value Theorem, Chapter 16). Start inside, near whichever end has the smaller $|f|$.
- **Use the problem's meaning.** For $\sqrt{2}$, anything between 1 and 2 is sensible; for a physical quantity, its order of magnitude is usually known.
- **Run a few steps of a safe method first** (bisection, below), then switch to Newton.

Avoid starting near points where $f'(x) = 0$ — turning points of the curve — for reasons we will see in step 10.

18.10.6. 6–7. Computing the next approximation, and repeating

Now do it by hand. **Find $\sqrt{2}$** using $f(x) = x^2 - 2$, whose derivative is $f'(x) = 2x$, starting from $x_0 = 1$ (since $1^2 = 1 < 2 < 4 = 2^2$, the root is between 1 and 2).

Substituting f and f' into the general formula first gives a specific iteration:

$$x_{n+1} = x_n - \frac{x_n^2 - 2}{2x_n}.$$

Iteration 1 ($n = 0$):

$$\begin{aligned} f(x_0) &= 1^2 - 2 = -1 \\ f'(x_0) &= 2(1) = 2 \\ \frac{f(x_0)}{f'(x_0)} &= \frac{-1}{2} = -0.5 \\ x_1 &= 1 - (-0.5) = 1.5 \end{aligned}$$

Iteration 2 ($n = 1$):

$$\begin{aligned} f(x_1) &= 1.5^2 - 2 = 2.25 - 2 = 0.25 \\ f'(x_1) &= 2(1.5) = 3 \\ \frac{f(x_1)}{f'(x_1)} &= \frac{0.25}{3} = 0.083\,333\ldots \\ x_2 &= 1.5 - 0.083\,333\ldots = 1.416\,666\ldots \quad \left(= \tfrac{17}{12}\right) \end{aligned}$$

Iteration 3 ($n = 2$). Working in fractions keeps it exact:

$$
\begin{aligned}
f(x_2) &= \left(\tfrac{17}{12}\right)^2 - 2 = \tfrac{289}{144} - \tfrac{288}{144} = \tfrac{1}{144} \approx 0.006\,944\,4 \\
f'(x_2) &= 2 \cdot \tfrac{17}{12} = \tfrac{17}{6} \approx 2.833\,333 \\
\frac{f(x_2)}{f'(x_2)} &= \tfrac{1}{144} \cdot \tfrac{6}{17} = \tfrac{1}{408} \approx 0.002\,450\,98 \\
x_3 &= \tfrac{17}{12} - \tfrac{1}{408} = \tfrac{578}{408} - \tfrac{1}{408} = \tfrac{577}{408} \approx 1.414\,215\,686
\end{aligned}
$$

Iteration 4 ($n = 3$):

$$
\begin{aligned}
f(x_3) &= \left(\tfrac{577}{408}\right)^2 - 2 = \tfrac{332\,929 - 332\,928}{166\,464} = \tfrac{1}{166\,464} \approx 6.007 \times 10^{-6} \\
f'(x_3) &= \tfrac{577}{204} \approx 2.828\,431 \\
\frac{f(x_3)}{f'(x_3)} &= \tfrac{1}{166\,464} \cdot \tfrac{204}{577} = \tfrac{1}{470\,832} \approx 2.124 \times 10^{-6} \\
x_4 &= \tfrac{577}{408} - \tfrac{1}{470\,832} = \tfrac{665\,857}{470\,832} \approx 1.414\,213\,562\,375
\end{aligned}
$$

Compare with the true value $\sqrt{2} = 1.414\,213\,562\,373\,095\ldots$:

n	x_n	error $\lvert x_n - \sqrt{2}\rvert$	correct digits
0	1	4.1×10^{-1}	0
1	1.5	8.6×10^{-2}	1
2	1.416 666 666 666 7	2.5×10^{-3}	3
3	1.414 215 686 274 5	2.1×10^{-6}	6
4	1.414 213 562 374 7	1.6×10^{-12}	12

Try it: Newton versus bisection for √2 .

Look at the last column: 1, 3, 6, 12. **The number of correct digits roughly doubles every step.** Four steps of arithmetic you could do on paper give twelve correct digits.

18.10.7. 8. Deciding when to stop

An algorithm needs a termination condition. Since we don't know the true root (if we did, we wouldn't be computing it), we stop on evidence available inside the loop:

- **Step size:** stop when $\lvert x_{n+1} - x_n\rvert < \varepsilon$ for a chosen **tolerance** ε (epsilon, the conventional letter for "a small positive number"). With quadratic convergence, once the step is tiny the error is tinier still: the step from x_3 to x_4 above was 2.1×10^{-6}, while x_4's actual error is about 10^{-12}.
- **Residual:** stop when $\lvert f(x_n)\rvert < \delta$. The **residual** $f(x_n)$ measures how nearly the equation is satisfied. On its own it can mislead: if the curve is very flat, $f(x)$ can be tiny far from the root.
- **Relative tolerance:** for roots of large or unknown magnitude, use $\lvert x_{n+1} - x_n\rvert < \varepsilon \lvert x_{n+1}\rvert$.

- **Iteration cap:** always stop after some maximum number of steps and report failure. Newton's method is not guaranteed to converge, and an unbounded loop is a bug.

Do not ask for more accuracy than the arithmetic can deliver. Double-precision floats carry about 16 significant digits (Chapter 32); a tolerance of 10^{-20} on a root near 1 can never be met.

18.10.8. The procedure

Example 12. Procedure — Newton's method

Input: a function $f(x)$, its derivative $f'(x)$, an initial estimate x_0, a tolerance ε, and a maximum number of iterations.

Repeat for $n = 0, 1, 2, \ldots$:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

until

$$|x_{n+1} - x_n| < \varepsilon.$$

(If $f'(x_n) = 0$, or the iteration cap is reached first, stop and report failure.)

Output: x_{n+1}, an approximate root.

Try it: Step Newton's cubic . ==== Worked example: Newton's own cubic

Solve $x^3 - 2x - 5 = 0$. This is the very equation Newton used to demonstrate his method around 1669; it has no rational root and one real root.

Set up. $f(x) = x^3 - 2x - 5$, $f'(x) = 3x^2 - 2$.

Initial value. Try integers: $f(2) = 8 - 4 - 5 = -1$ and $f(3) = 27 - 6 - 5 = 16$. The sign change puts a root between 2 and 3, and $|f(2)|$ is much smaller, so take $x_0 = 2$.

Iteration 1:

$$\begin{aligned} f(2) &= 8 - 4 - 5 = -1 \\ f'(2) &= 3(4) - 2 = 10 \\ x_1 &= 2 - \frac{-1}{10} = 2 + 0.1 = 2.1 \end{aligned}$$

Iteration 2:

$$
\begin{aligned}
2.1^3 &= 9.261 \\
f(2.1) &= 9.261 - 4.2 - 5 = 0.061 \\
f'(2.1) &= 3(4.41) - 2 = 13.23 - 2 = 11.23 \\
\frac{f(2.1)}{f'(2.1)} &= \frac{0.061}{11.23} = 0.005\,431\,879\ldots \\
x_2 &= 2.1 - 0.005\,431\,879 = 2.094\,568\,121
\end{aligned}
$$

Iteration 3:

$$
\begin{aligned}
x_2^2 &= 4.387\,215\,614, \qquad x_2^3 = 9.189\,321\,965 \\
f(x_2) &= 9.189\,321\,965 - 4.189\,136\,242 - 5 = 0.000\,185\,723 \\
f'(x_2) &= 3(4.387\,215\,614) - 2 = 11.161\,646\,84 \\
\frac{f(x_2)}{f'(x_2)} &= \frac{0.000\,185\,723}{11.161\,646\,84} = 0.000\,016\,639\,41 \\
x_3 &= 2.094\,568\,121 - 0.000\,016\,639 = 2.094\,551\,482
\end{aligned}
$$

Iteration 4: $f(x_3) \approx 1.74 \times 10^{-9}$, $f'(x_3) \approx 11.1614$, correction $\approx 1.56 \times 10^{-10}$, giving $x_4 = 2.094\,551\,481\,542\,3$.

Stop? The last step was 1.6×10^{-10}; with $\varepsilon = 10^{-8}$ we stop. The root is $x \approx 2.094\,551\,48$. The corrections shrank $0.1 \to 5.4 \times 10^{-3} \to 1.7 \times 10^{-5} \to 1.6 \times 10^{-10}$ — again, the exponent roughly doubles.

18.10.9. Implementation in Python

The procedure translates almost line for line. We record every iterate so that we can study convergence.

PYTHON

```python
def newton(f, df, x0, tol=1e-12, max_iter=50):
    """Return (root, history). Raises if it fails to converge."""
    x = x0
    history = [x]
    for _ in range(max_iter):
        fx, dfx = f(x), df(x)
        if dfx == 0:
            raise ZeroDivisionError(f"f'(x) = 0 at x = {x}; tangent is flat")
        x_new = x - fx / dfx
        history.append(x_new)
        if abs(x_new - x) < tol:
            return x_new, history
        x = x_new
    raise RuntimeError(f"no convergence after {max_iter} steps: {history[-3:]}")


root, hist = newton(lambda x: x**2 - 2, lambda x: 2 * x, 1.0)
for n, x in enumerate(hist):
    print(f"x_{n} = {x:.15f}   error = {abs(x - 2**0.5):.1e}")

root, hist = newton(lambda x: x**3 - 2*x - 5, lambda x: 3*x**2 - 2, 2.0)
print(root, len(hist) - 1, "steps")

try:
    newton(lambda x: x**3 - 2*x + 2, lambda x: 3*x**2 - 2, 0.0)
except RuntimeError as e:
    print(e)
```

```
x_0 = 1.000000000000000   error = 4.1e-01
x_1 = 1.500000000000000   error = 8.6e-02
x_2 = 1.416666666666667   error = 2.5e-03
x_3 = 1.414215686274510   error = 2.1e-06
x_4 = 1.414213562374690   error = 1.6e-12
x_5 = 1.414213562373095   error = 0.0e+00
x_6 = 1.414213562373095   error = 2.2e-16
2.0945514815423265 5 steps
no convergence after 50 steps: [0.0, 1.0, 0.0]
```

Three things to notice. The hand calculation and the program agree digit for digit. After x_5 the error is at the level of floating-point rounding — x_6 differs from x_5 by one unit in the last place, and that tiny step is what triggers the stop. And the last call, which we will explain shortly, never converges at all: the iteration cap turned an infinite loop into a clear error message.

18.10.10. 9. When Newton's method works well: quadratic convergence

Why do the digits double? Let r be the root and write the error as $e_n = x_n - r$. Expand f around x_n using a Taylor polynomial of degree 1 plus its error term (Section 18.8), evaluated at the root:

$$0 = f(r) = f(x_n) + f'(x_n)(r - x_n) + \tfrac{1}{2} f''(\xi)(r - x_n)^2$$

for some ξ between x_n and r. Divide by $f'(x_n)$ and rearrange:

$$0 = \frac{f(x_n)}{f'(x_n)} + r - x_n + \frac{f''(\xi)}{2f'(x_n)}\,e_n^2$$

$$x_n - \frac{f(x_n)}{f'(x_n)} - r = \frac{f''(\xi)}{2f'(x_n)}\,e_n^2$$

$$e_{n+1} = \frac{f''(\xi)}{2f'(x_n)}\,e_n^2.$$

The new error is proportional to the *square* of the old one. If the error is 10^{-3}, the next is about $C \times 10^{-6}$, then $C^3 \times 10^{-12}$: the exponent doubles. This is **quadratic convergence**.

Check it on $\sqrt{2}$: $C \approx \frac{f''(r)}{2f'(r)} = \frac{2}{2\cdot 2\sqrt{2}} \approx 0.3536$. From $e_3 = 2.124 \times 10^{-6}$, the prediction is $e_4 \approx 0.3536 \times (2.124 \times 10^{-6})^2 = 1.595 \times 10^{-12}$, and the measured error was 1.595×10^{-12}.

The formula also explains what "works well" requires: $f'(r) \neq 0$ (a **simple** root, so C is finite), f'' not wild, and x_0 close enough that $|Ce_0| < 1$. Under those conditions Newton's method is the fastest practical root finder there is.

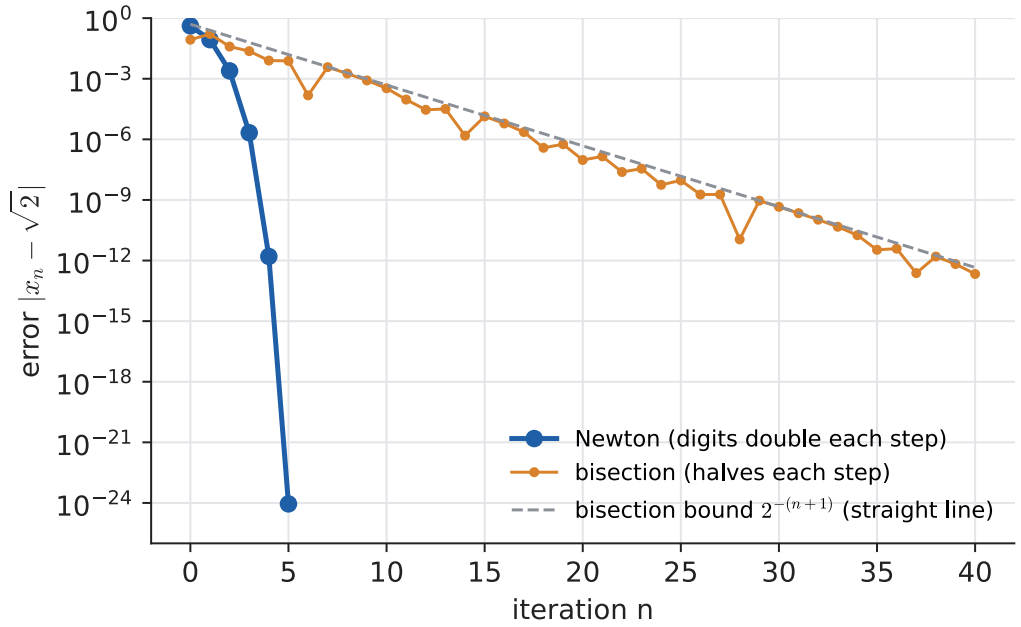


Figure 29. Error per iteration on a logarithmic scale. Newton's error curve bends downward ever more steeply (digits double). Bisection's actual error zigzags, but it always stays under its guaranteed bound $2^{-(n+1)}$, a straight line on this scale (one binary digit per step). Newton reaches about 10^{-12} in 4 steps (1.6×10^{-12}); bisection needs about 40.



Order of convergence

A method has **order of convergence** p if $|e_{n+1}| \approx C|e_n|^p$ for large n. $p = 1$ is **linear convergence** (a fixed number of digits gained per step — bisection gains one binary digit); $p = 2$ is **quadratic** (digits double — Newton at a simple root). The secant method has $p = \frac{1+\sqrt{5}}{2} \approx 1.618$, the golden ratio.

Try it: Race the two methods .

18.10.11. 10. When Newton's method fails

Quadratic convergence is a *local* promise: it holds once you are close enough. Far from a root, or at a badly behaved one, the tangent line is a poor model of the curve, and following it can lead anywhere. These are not curiosities; you will meet every one of them in practice.

Oscillation: $f(x) = x^3 - 2x + 2$ from $x_0 = 0$. Here $f'(x) = 3x^2 - 2$. *Try it:* Watch the 0–1 cycle .

$$x_0 = 0: \quad f(0) = 2, \quad f'(0) = -2, \quad x_1 = 0 - \frac{2}{-2} = 1$$

$$x_1 = 1: \quad f(1) = 1 - 2 + 2 = 1, \quad f'(1) = 3 - 2 = 1, \quad x_2 = 1 - \frac{1}{1} = 0$$

We are back where we started; the iteration cycles $0, 1, 0, 1, \ldots$ forever. Worse, the cycle *attracts* nearby starts: from $x_0 = 0.1$ the iterates are drawn into the same 0–1 cycle. The only real root is near -1.769, on the other side of a hump. Start at $x_0 = -2$ instead and Newton gives $-1.8, \ -1.76995, \ -1.7692927, \ -1.769292354$ — converged in four steps.

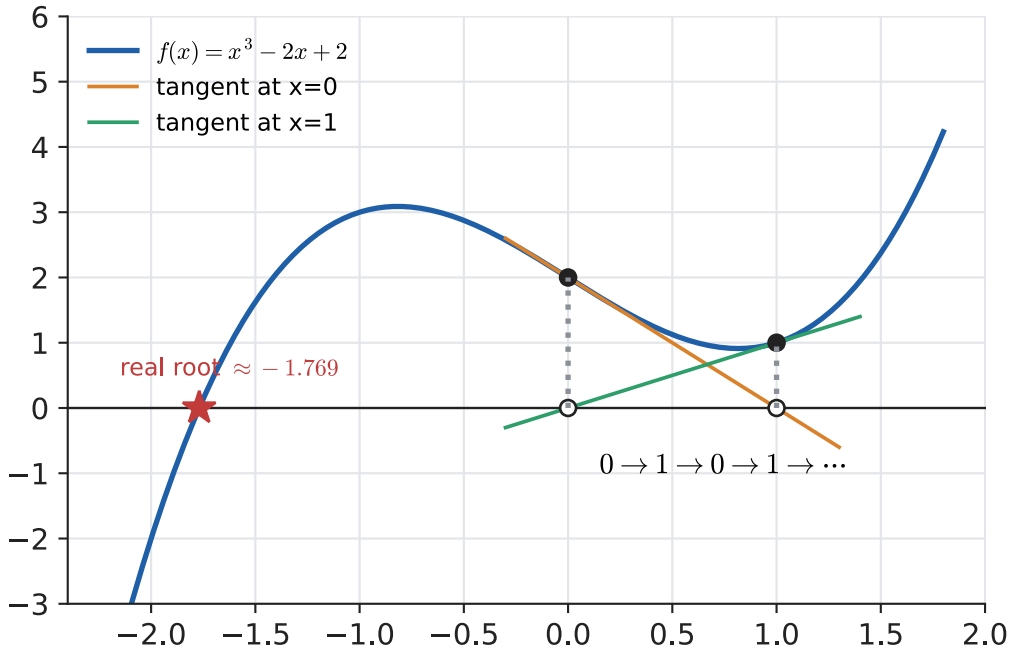


Figure 30. Newton's method cycling on $x^3 - 2x + 2$: the tangent at 0 points to 1, and the tangent at 1 points back to 0. The real root (star) is never approached.

Divergence: the iterates run away. For $f(x) = \sqrt[3]{x} = x^{1/3}$ (root at 0), $f'(x) = \frac{1}{3}x^{-2/3}$, and *Try it:* Find the arctan threshold ; Newton diverging on ∛x .

$$x_{n+1} = x_n - \frac{x_n^{1/3}}{\frac{1}{3}x_n^{-2/3}} = x_n - 3x_n = -2x_n.$$

Every step *doubles* the distance from the root, from any nonzero start: $0.5, -1, 2, -4, \ldots$. The curve is so steep at its root (vertical tangent) that the tangents from elsewhere overshoot. A less contrived example: $f(x) = \arctan x$ from $x_0 = 1.5$ gives $-1.694, \ 2.321, \ -5.114, \ 32.30, \ -1575.3, \ldots$ —

the curve flattens out away from the root, so each tangent shoots further. (From $x_0 = 1$ it converges; the boundary is near $x_0 \approx 1.39$.)

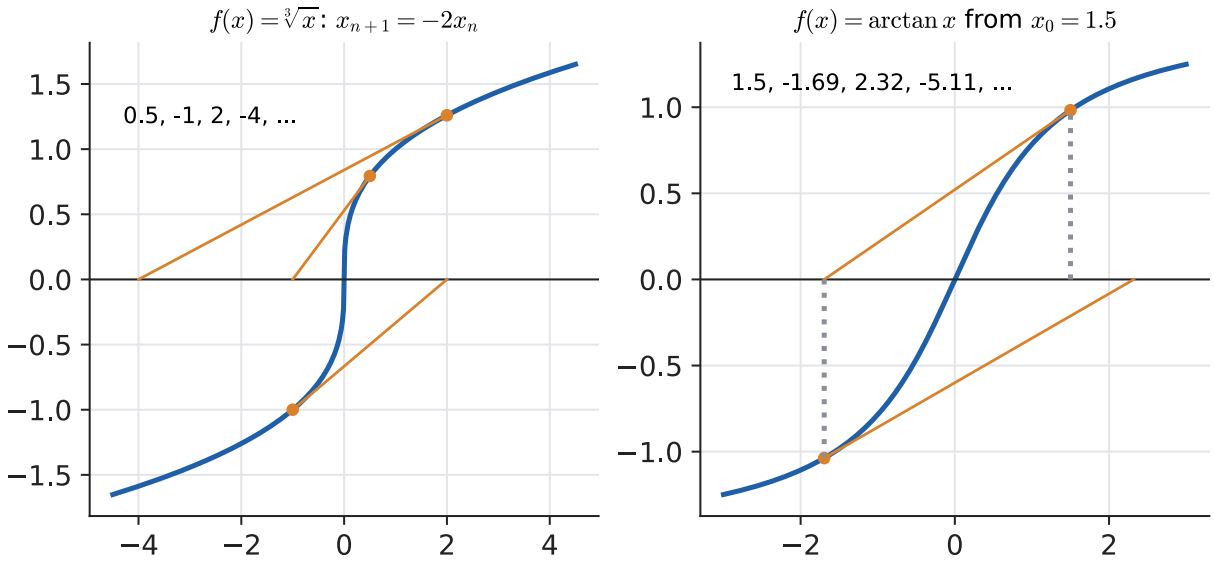


Figure 31. Two ways to diverge. Left: for $\sqrt[3]{x}$ each tangent overshoots to twice the distance on the other side. Right: for $\arctan x$ from $x_0 = 1.5$, the flattening curve makes each tangent land further out.

Flat tangent: $f'(x_n) \approx 0$. The correction $f(x_n)/f'(x_n)$ divides by the slope; near a turning point the slope is tiny and the step is huge. For $x^2 - 2$ from $x_0 = 0.01$: *Try it:* The flat-tangent leap .

$$x_1 = 0.01 - \frac{0.0001 - 2}{0.02} = 0.01 + 99.995 = 100.005.$$

Here the parabola is kind and the iterates crawl back ($50.01, 25.03, 12.55, 6.36, 3.34, 1.97, 1.49, 1.416, \ldots$), costing 11 steps instead of 5. On a function with several roots the giant leap usually lands near a *different* root, or somewhere the method then diverges. Exactly at $f'(x_n) = 0$ the tangent is horizontal, never meets the axis, and the formula divides by zero.

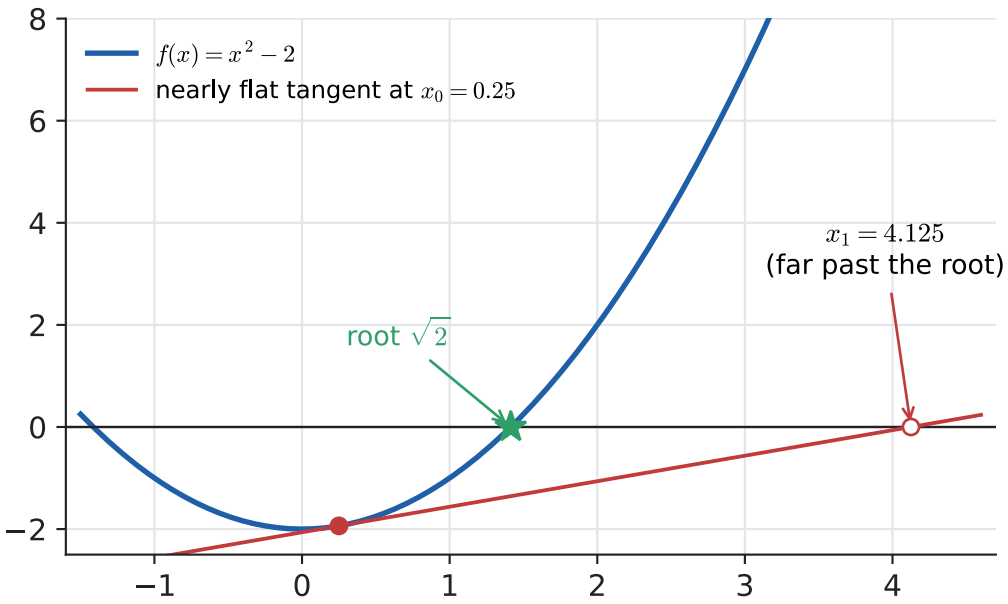


Figure 32. Starting near a turning point, a gentler version of the example above: the nearly flat tangent at $x_0 = 0.25$ sends x_1 out to 4.125 (from $x_0 = 0.01$ it would land at 100, off the picture).

Wrong root, and sensitive dependence on x_0. $f(x) = x^3 - x = x(x-1)(x+1)$ has roots $-1, 0, 1$. You might expect Newton to find the nearest one. Not so: starting from 0.44 reaches 0; from 0.45 it reaches -1; from 0.46 it reaches $+1$; from 0.5 it reaches -1 again. Near $x = \pm 1/\sqrt{3} \approx \pm 0.577$, where $f' = 0$, tiny tangents throw the iterate across the graph. The set of starting values that lead to a given root is that root's **basin of attraction**, and the basins interleave in ever finer bands near $\pm 1/\sqrt{5} \approx \pm 0.447$. *Try it:* Zoom into the basins .

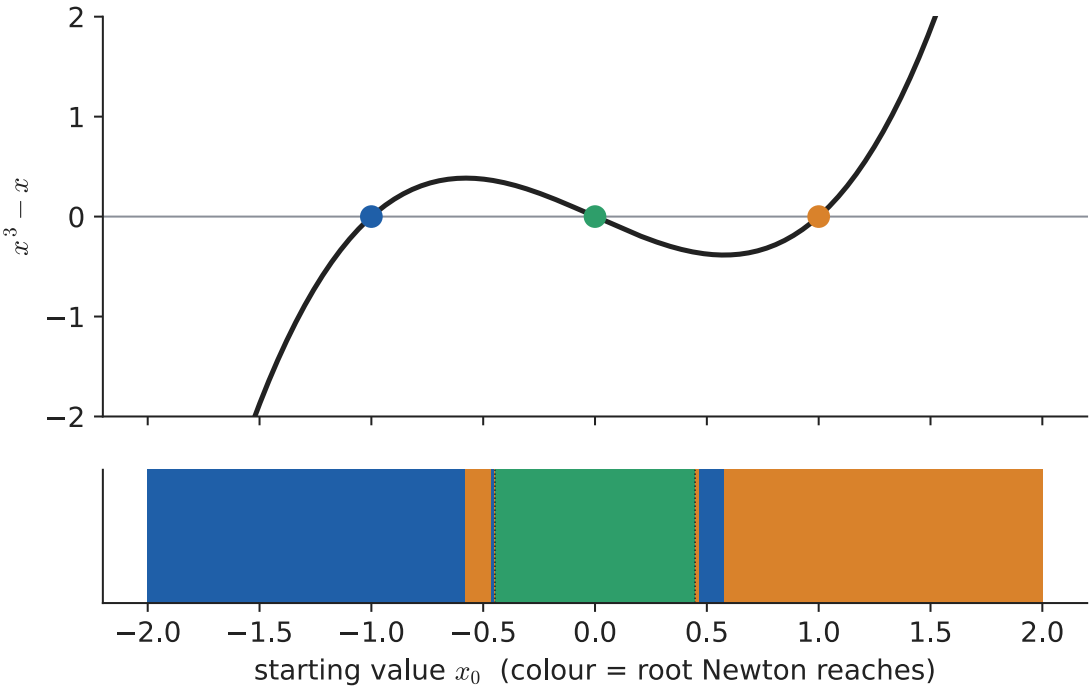


Figure 33. Basins of attraction for $x^3 - x$. Each starting value on the bottom strip is coloured by the root Newton's method converges to. Most of the line behaves sensibly, but near ± 0.45 the colours alternate in thinner and thinner bands.

Slow convergence at a repeated root. For $f(x) = (x-1)^2$, the root $x = 1$ is a double root where $f'(1) = 0$ too. The iteration becomes $x_{n+1} = x_n - \frac{(x_n-1)^2}{2(x_n-1)} = x_n - \frac{x_n-1}{2}$: the error merely halves each step. Newton degrades to linear convergence — no better than bisection — whenever the curve touches the axis instead of crossing it. *Try it:* Newton at a double root .



Newton's method: checklist of failure modes

- **Divergence** — the iterates run off to infinity ($\sqrt[3]{x}$, $\arctan x$ from too far out).
- **Oscillation** — they cycle forever ($x^3 - 2x + 2$ from 0).
- **Wrong root** — they converge, but to a root you didn't want; basins can be intricate.
- $f' \approx 0$ — near a turning point, a huge, unreliable step; at $f' = 0$, division by zero.
- **Multiple roots** — convergence is merely linear.
- **Dependence on x_0** — everything above depends on the start. A good initial guess is half the algorithm.

Defences: plot first; bracket the root with a sign change; cap the iterations; reject steps that leave the bracket and fall back to bisection (that is what robust library routines do); check the final residual $|f(x)|$.

18.10.12. Alternatives and choosing a method

Newton's method is one of several ways to solve $f(x) = 0$. The most important companion is **bisection**.

Bisection. If $f(a)$ and $f(b)$ have opposite signs, a root lies in $[a, b]$. Evaluate the midpoint $m = \frac{a+b}{2}$; whichever half still has a sign change contains a root; keep that half and repeat.

```python
def bisection(f, a, b, tol=1e-12):
    fa = f(a)
    if fa * f(b) > 0:
        raise ValueError("f(a) and f(b) must have opposite signs")
    steps = 0
    while b - a > tol:
        m = (a + b) / 2
        fm = f(m)
        if fa * fm <= 0:
            b = m
        else:
            a, fa = m, fm
        steps += 1
    return (a + b) / 2, steps


print(bisection(lambda x: x**2 - 2, 1.0, 2.0))
```

```
(1.414213562372879, 40)
```

Forty steps for twelve digits, against Newton's four or five. But bisection *cannot* fail once you have a bracket: the interval halves every time, so after n steps the error is at most $(b - a)/2^n$. It is binary search (Volume 1, Chapter 12) applied to a continuous function.

Secant method. Replace $f'(x_n)$ by the slope of the secant through the last two iterates: $x_{n+1} = x_n - f(x_n)\dfrac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$. No derivative needed; order ≈ 1.618; same failure modes as Newton.

Closed-form algebra. If a formula exists, use it. $x^2 - 5x + 6 = 0$ practically begs to be factored as $(x - 2)(x - 3) = 0$; running Newton's method on it would be rather silly.

Method	Needs	Speed	Reliability
Algebra / formula	a solvable form	instant, exact	perfect, when available
Bisection	continuous f, a sign-change bracket	linear: 1 bit per step (~3.3 steps per decimal digit)	guaranteed once bracketed
Newton	f and f', a good x_0	quadratic: digits double	can diverge, cycle, or jump roots
Secant	f, two starting points	order ~1.6, no derivative cost	like Newton
Hybrid (e.g. Brent's method)	a bracket	near-Newton speed	guaranteed

Choosing: use algebra when it applies. When it doesn't: if you have a good starting guess and a cheap derivative (or need to solve many similar equations — e.g. a square root for every pixel), use Newton. If you know nothing about where the root is, or reliability matters more than speed, bracket and bisect. In production code, use a hybrid: library routines such as Brent's method take a fast Newton- or secant-like step when it stays inside the bracket and a bisection step when it doesn't, getting both speed and a guarantee. Understanding the two pure methods is how you understand — and debug — the hybrid.

18.10.13. Where you'd use this

- **Square roots, division and reciprocals in hardware and software.** Calculators and CPUs compute $\sqrt{a}$ and $1/a$ by Newton iterations from a table-lookup first guess.
- **Finance: internal rate of return.** You invest £1000 and receive £400 at the end of each of the next three years. The **internal rate of return** is the interest rate r at which those payments are worth exactly £1000 today: $f(r) = \frac{400}{1+r} + \frac{400}{(1+r)^2} + \frac{400}{(1+r)^3} - 1000 = 0$, a cubic in $1/(1+r)$. Newton from $r_0 = 0.1$ gives $r_1 = 0.096997$, $r_2 = 0.0970103$: an IRR of about 9.70%. Spreadsheet `IRR` functions do exactly this — and print an error when the iteration fails to converge.
- **Engineering and physics:** implicit time-stepping in simulations solves a nonlinear equation at every step, by Newton's method.
- **Optimisation:** applied to $g'(x) = 0$, Newton's method finds stationary points: $x_{n+1} = x_n - g'(x_n)/g''(x_n)$. Its many-variable form uses the matrix of second derivatives (Chapter 20).
- **Graphics and geometry:** ray tracers find where a ray hits a curved surface by solving $f(t) = 0$ along the ray.

Computer-science connection: division without dividing

How does a chip compute $1/a$ if division is the operation you lack? Apply Newton's method to $f(x) = \frac{1}{x} - a$, whose root is $1/a$. With $f'(x) = -\frac{1}{x^2}$:

$$x_{n+1} = x_n - \frac{\frac{1}{x_n} - a}{-\frac{1}{x_n^2}} = x_n + x_n^2\left(\frac{1}{x_n} - a\right) = x_n(2 - a x_n).$$

Only multiplications and a subtraction! Starting from a lookup-table guess good to 8 bits, three iterations give 16, 32, then 64 correct bits. Variants of this are used for floating-point division in many processors and GPUs. The same trick on $f(x) = 1/x^2 - a$ gives the famous "fast inverse square root" from the *Quake III* engine: a bit-level hack produced a rough x_0, and one Newton step $x_1 = x_0(1.5 - 0.5\,a\,x_0^2)$ polished it, for normalising millions of lighting vectors per second.



Older than Newton, named after Newton

For $f(x) = x^2 - a$, Newton's formula simplifies to $x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$: average your guess with a divided by your guess. This is the **Babylonian** (or **Heron's**) method, and a Babylonian clay tablet, YBC 7289 (around 1700 BCE), records $\sqrt{2} \approx 1.414\,213$ in base 60 — accurate to about one part in a million. Newton (around 1669) described his method on the cubic $x^3 - 2x - 5 = 0$ as a sequence of polynomial substitutions, without derivatives; Joseph Raphson published the simpler iterative form in 1690; and the tangent-line formula we use came later still, with Thomas Simpson in 1740.



Newton's method explorer

Adjust the starting value x_0 for Newton's method with a slider and observe which root it reaches. The curve is drawn with each tangent line and each new iterate animated, and a side panel shows a table of x_n, $f(x_n)$, $f'(x_n)$ and the correction. Preset functions: $x^2 - 2$ (calm), $x^3 - 2x + 2$ (find the start that cycles), $\arctan x$ (find the divergence threshold near 1.39), and $x^3 - x$ (colour the starting line by which root wins and zoom into the interleaved basins). Toggle "bisection" to race the two methods on the same function and plot their errors on a log scale. Extension: in the complex plane (Volume 1, Chapter 15), colour each starting point z_0 by the root of $z^3 - 1$ it reaches, and watch a fractal appear.

Try it: Colour x³ − x basins · Open the Newton explorer · The z³ − 1 fractal

US curriculum

Newton's method appears in some **AP Calculus** courses as an extension of linear approximation (it is not on the current AP exam), and routinely in college **Calculus I** and in introductory **numerical analysis**. Bisection relies on the Intermediate Value Theorem, which is in AP Calculus AB.

18.11. A teaser: gradient descent

Newton's method finds where $f(x) = 0$. Optimisation needs where $g'(x) = 0$. A more modest cousin of Newton's method does this without second derivatives, and it is the engine of modern machine learning.

Idea. To minimise g, stand at x_n and look at the slope. If $g'(x_n) > 0$, downhill is to the left; if $g'(x_n) < 0$, downhill is to the right. So step against the slope, by an amount proportional to it:

$$x_{n+1} = x_n - \eta\, g'(x_n),$$

where η (eta) is a small positive **step size**, called the **learning rate** in machine learning. Steep slope: big step. Flat ground (near a minimum): small steps.

Example. Minimise $g(x) = (x-3)^2 + 1$, with $g'(x) = 2(x-3)$, from $x_0 = 0$ with $\eta = 0.2$: *Try it:* Step down the parabola .

$$\begin{aligned} x_1 &= 0 - 0.2 \cdot 2(0-3) = 0 + 1.2 = 1.2 \\ x_2 &= 1.2 - 0.2 \cdot 2(1.2-3) = 1.2 + 0.72 = 1.92 \\ x_3 &= 1.92 - 0.2 \cdot 2(1.92-3) = 1.92 + 0.432 = 2.352 \end{aligned}$$

PYTHON

```python
def gradient_descent(df, x0, eta=0.2, steps=8):
    x = x0
    for n in range(steps):
        x = x - eta * df(x)
        print(f"step {n + 1}: x = {x:.4f}")
    return x

gradient_descent(lambda x: 2 * (x - 3), 0.0)
```

```
step 1: x = 1.2000
step 2: x = 1.9200
step 3: x = 2.3520
step 4: x = 2.6112
step 5: x = 2.7667
step 6: x = 2.8600
step 7: x = 2.9160
step 8: x = 2.9496
```

The distance to the minimum at 3 is multiplied by $1 - 2\eta = 0.6$ each step: linear convergence, much slower than Newton. (Newton on g', namely $x - g'(x)/g''(x) = x - (x-3) = 3$, lands on the minimum in one step, because it uses the curvature g'' to choose the step.) Too large an η — try

$\eta = 1.1$ — and the iterates overshoot and diverge, the gradient-descent cousin of Newton's failure modes. *Try it:* Watch η = 1.1 diverge .

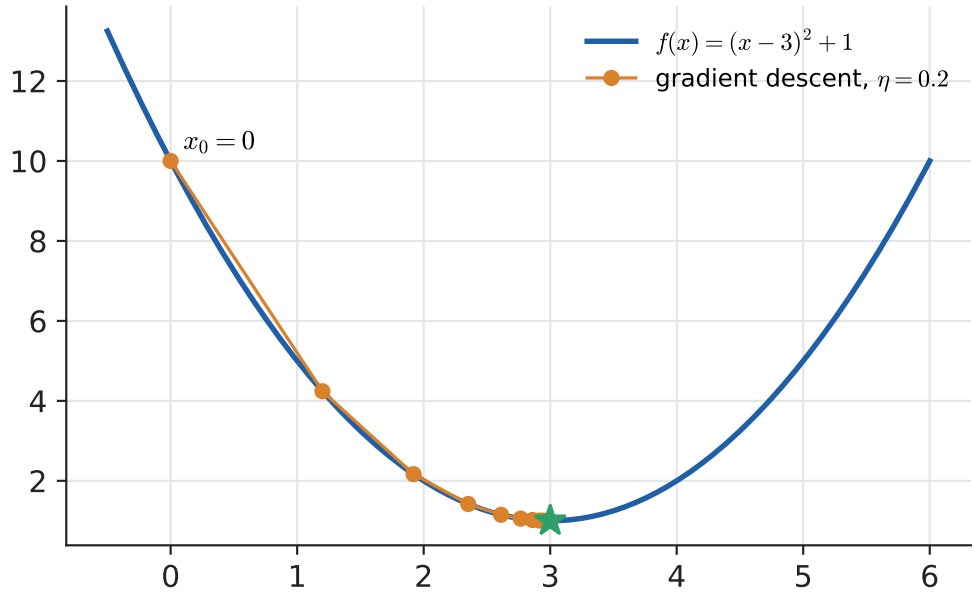


Figure 34. Gradient descent on a parabola: each step moves downhill by an amount proportional to the slope, so the steps shrink as the minimum approaches.

Why use the slower method? Because with a million parameters, the "second derivative" becomes a million-by-million matrix, too big to form, let alone invert; the first derivatives (the **gradient**, Chapter 20) can be computed for the cost of a few evaluations of the function by automatic differentiation (Chapter 17). Training a neural network is gradient descent, with clever choices of η, on a function of millions of variables (Volume 3, Chapter 33).

Computer-science connection: Newton vs gradient descent in practice

The trade-off between these two methods runs through all of numerical optimisation. Newton-type methods (and quasi-Newton methods such as L-BFGS, which build a cheap approximation to the second derivatives from successive gradients) dominate when the number of variables is modest — curve fitting, logistic regression, physics solvers. Gradient descent and its variants (momentum, Adam) dominate deep learning, where each step must be cheap and the data are noisy. Both are "follow the local linear (or quadratic) model, then repeat" — the idea of this whole chapter.

18.12. Exercises: Newton's method

1. **(Recognition)** Write down the Newton iteration for $f(x) = x^3 - 10$, and simplify it.
2. **(Application)** Starting from $x_0 = 2$, perform three Newton iterations for $\sqrt[3]{10}$, showing $f(x_n)$, $f'(x_n)$ and the correction at each step.
3. **(Application)** Solve $\cos x = x$ with Newton's method from $x_0 = 1$. Give x_1 and x_2 with all intermediate values.

4. **(Problem solving)** Explain, with a sketch, why Newton's method for $f(x) = x^3 - 2x + 2$ from $x_0 = 0.5$ wanders before converging (iterates: $1.4, 0.899, -1.289, -2.106, -1.829, -1.772, \ldots$). Where is $f'(x) = 0$?

5. **(Problem solving)** For $f(x) = x^2 - a$ ($a > 0$) show that Newton's method is the Babylonian rule $x_{n+1} = \frac{1}{2}(x_n + a/x_n)$, and that from any $x_0 > 0$ the iterates after the first are all $\geq \sqrt{a}$. (Hint: $x_{n+1} - \sqrt{a} = (x_n - \sqrt{a})^2/(2x_n)$.)

6. **(Problem solving)** You need to solve $e^x = 3x$ in a program that runs a million times per second, and also once in a one-off script. Which method would you choose in each case and why?

▼ **Answers**

1. $f'(x) = 3x^2$, so $x_{n+1} = x_n - \frac{x_n^3 - 10}{3x_n^2} = \frac{2x_n^3 + 10}{3x_n^2} = \frac{2}{3}x_n + \frac{10}{3x_n^2}$.

2. $n = 0$: $f(2) = 8 - 10 = -2$, $f'(2) = 12$, correction $-2/12 = -0.16667$, $x_1 = 2.16667$. $n = 1$: $f(x_1) = 10.17130 - 10 = 0.17130$, $f'(x_1) = 14.08333$, correction 0.012163, $x_2 = 2.15450$. $n = 2$: $f(x_2) = 0.000960$, $f'(x_2) = 13.92566$, correction 0.0000689, $x_3 = 2.1544347$. (True: $2.15443469\ldots$)

3. $f(x) = \cos x - x$, $f'(x) = -\sin x - 1$. $n = 0$: $f(1) = 0.540302 - 1 = -0.459698$, $f'(1) = -0.841471 - 1 = -1.841471$, correction $-0.459698/-1.841471 = 0.249636$, $x_1 = 1 - 0.249636 = 0.750364$. $n = 1$: $f(x_1) = 0.731440 - 0.750364 = -0.018923$, $f'(x_1) = -0.681905 - 1 = -1.681905$, correction 0.011251, $x_2 = 0.739113$. Converges to 0.7390851.

4. $f'(x) = 3x^2 - 2 = 0$ at $x = \pm\sqrt{2/3} \approx \pm 0.816$. Starting at 0.5 the curve is near its local minimum at 0.816, where the tangent is shallow; the tangent at 0.5 (slope -1.25) sends the iterate right to 1.4, then the iterate at 0.899 is close to the turning point and its shallow tangent throws it far left, past the root, to -1.289. From there on it is on the steep left branch and converges quickly to -1.7693.

5. $x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{2x_n^2 - x_n^2 + a}{2x_n} = \frac{x_n^2 + a}{2x_n} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$. Then $x_{n+1} - \sqrt{a} = \frac{x_n^2 - 2\sqrt{a}\,x_n + a}{2x_n} = \frac{(x_n - \sqrt{a})^2}{2x_n} \geq 0$ for $x_n > 0$. So every iterate after x_0 is at or above $\sqrt{a}$, and (being above) each further step moves down toward it — no failure is possible for $x_0 > 0$.

6. Graph first: $e^x - 3x$ has two roots (near 0.62 and 1.51), because $f(0) = 1 > 0$, $f(1) = e - 3 < 0$, $f(2) = e^2 - 6 > 0$. For the high-volume program: Newton ($f'(x) = e^x - 3$ is cheap) from a fixed good starting value inside the right bracket, with a cap and a fallback — e.g. from $x_0 = 0.5$: $0.6100, 0.6190, \ldots$. Avoid starting near $x = \ln 3 \approx 1.099$, where $f' = 0$. For the one-off script, robustness beats speed: bisection on $[0, 1]$ (or a library hybrid such as Brent's method) is simplest to get right.

18.13. Interactive tools

Browser tools on the companion website that go with this chapter. Each example link in the chapter opens one of them with that example already set up.

- Calculus, visually
- Symbolic algebra
- Graphing calculator
- Root finding, step by step
- Surfaces, gradients and gradient descent

18.14. Mastery check

1. Find and classify the stationary points of $f(x) = 2x^3 - 3x^2 - 12x + 1$, and give its inflection point.
2. An open-topped box with a square base must hold $32\,000\ \text{cm}^3$. Find the dimensions that use the least material.
3. Use a linear approximation to estimate $(1.02)^{10}$. Is the estimate high or low?
4. Evaluate $\displaystyle\lim_{x \to 0} \frac{e^{2x} - 1}{x}$.
5. Write the degree-3 Taylor polynomial of $\sin x$ about 0 and use it to estimate $\sin 0.2$.
6. Derive Newton's formula from the tangent line, then perform two iterations for $f(x) = x^2 - 5$ from $x_0 = 2$.
7. Newton's method on $x^3 - 2x + 2$ from $x_0 = 0$ does not converge. Show what happens in two steps, and suggest two different fixes.

Decision rule: All correct without hesitation → move on to Chapter 19. Stuck on 1–2 → reread Section 18.1; on 2 → Section 18.3; on 3–5 → Section 18.6, Section 18.7 and Section 18.8; on 6–7 → Section 18.10 and Section 18.10.11. Stuck on more → work the section exercises first.

▼ **Mastery check answers**

1. $f'(x) = 6x^2 - 6x - 12 = 6(x - 2)(x + 1)$: stationary at $x = -1$ and $x = 2$.
   $f''(x) = 12x - 6$. $f''(-1) = -18 < 0$: local max $f(-1) = -2 - 3 + 12 + 1 = 8$.
   $f''(2) = 18 > 0$: local min $f(2) = 16 - 12 - 24 + 1 = -19$. Inflection where $12x - 6 = 0$: $x = \frac{1}{2}$, $f(\frac{1}{2}) = \frac{1}{4} - \frac{3}{4} - 6 + 1 = -5.5$.
2. Base side x, height h; constraint $x^2 h = 32\,000$, so $h = 32\,000/x^2$. Material $S = x^2 + 4xh = x^2 + 128\,000/x$.
   $S'(x) = 2x - 128\,000/x^2 = 0 \Rightarrow x^3 = 64\,000 \Rightarrow x = 40$. $S''(x) = 2 + 256\,000/x^3 > 0$: minimum. $h = 32\,000/1600 = 20$. Box $40 \times 40 \times 20$ cm, using $4800\ \text{cm}^2$.
3. $f(x) = x^{10}$ at $a = 1$: $f(1) = 1$, $f'(1) = 10$, so $(1.02)^{10} \approx 1 + 10(0.02) = 1.2$. True value 1.21899: the estimate is low, because x^{10} is concave up there (tangent below the curve).
4. Form $\frac{0}{0}$. L'Hôpital: $\lim_{x \to 0} \frac{2e^{2x}}{1} = 2$. (Or: $e^{2x} \approx 1 + 2x$, so the ratio $\approx 2x/x = 2$.)

5. $T_3(x) = x - \frac{x^3}{6}$. $\sin 0.2 \approx 0.2 - \frac{0.008}{6} = 0.2 - 0.001\,333 = 0.198\,667$ (true: $0.198\,669$).

6. Tangent at x_n: $y = f(x_n) + f'(x_n)(x - x_n)$; setting $y = 0$ gives $x_{n+1} = x_n - f(x_n)/f'(x_n)$. With $f'(x) = 2x$: $n = 0$: $f(2) = -1$, $f'(2) = 4$, $x_1 = 2 + 0.25 = 2.25$. $n = 1$: $f(2.25) = 5.0625 - 5 = 0.0625$, $f'(2.25) = 4.5$, correction $0.013\,889$, $x_2 = 2.236\,111$ (true $\sqrt{5} = 2.236\,068$).

7. $x_1 = 0 - \frac{2}{-2} = 1$; $x_2 = 1 - \frac{1}{1} = 0$: a 2-cycle. Fixes: (a) choose a better x_0 from a sketch or sign change — $f(-2) = -2 < 0 < 2 = f(0)$, so start at $x_0 = -2$, which converges to -1.76929 in four steps; (b) use bisection on $[-2, 0]$ (or a bracketed hybrid), which cannot cycle.

Chapter 19. The Integral

Chapter map. The derivative answered "how fast is this changing right now?" This chapter answers the reverse question: "if I know how fast something changes at every instant, how much has it changed in total?" We start with a car whose speed keeps changing and ask how far it travelled. That question turns into the area under a curve. We approximate the area with rectangles (Riemann sums), push the approximation to a limit (the definite integral), and then meet the result that ties all of calculus together: the Fundamental Theorem of Calculus, which says integration is differentiation run backwards. After that come the techniques (antiderivatives, substitution, integration by parts), the applications (areas, averages, volumes, work, probability), integrals over infinite ranges, numerical integration for when the algebra gives up, and a first look at differential equations, which is how physics engines and population models actually run.

Why this chapter matters: whenever you know a *rate* and want a *total*, you are integrating. Your electricity bill is the integral of power over the month; a fitness tracker's distance is the integral of speed; a drug's total exposure is the area under its concentration curve; a cloud bill is the integral of usage over time; a game advances its world by integrating velocity every frame. Probability for continuous quantities is defined as area, which is why the integral underpins statistics and machine learning too.

Where this chapter sits

You need:

- Limits, in particular the idea of a quantity approaching a value as a step size shrinks (Chapter 16).
- Derivatives and the main rules: power, product and chain rules, and the derivatives of $\sin$, $\cos$, e^x and $\ln x$ (Chapter 17).
- Summation and sequences, including Σ notation (Volume 1, Chapter 14). It is re-explained below.
- Exponentials and logarithms (Volume 1, Chapter 12) and basic trigonometry (Volume 1, Chapter 13).

This chapter unlocks:

- Multivariable calculus and double integrals (Chapter 20).
- Continuous probability distributions, where probabilities are areas (Volume 3, Chapter 30).
- Numerical methods: quadrature, simulation and ODE solvers (Chapter 32).
- The mathematics of machine learning and physics simulation (Volume 3, Chapter 33).

19.1. The problem: how far did the car go?

19.1.1. Constant speed is easy

If a car drives at a steady 60 mph for 2 hours it covers $60 \times 2 = 120$ miles. Draw that on a graph with time across and speed up, and the journey is a horizontal line at height 60 running from $t = 0$ to $t = 2$. The distance, 120, is the **area of the rectangle** under that line: height (speed) times width (time).

Now let the speed change in jumps. Say 30 mph for the first hour, 60 mph for the next 1.5 hours, then 20 mph for the last half hour. Each piece is a rectangle, and the total distance is the sum of their areas:

$$30 \times 1 \;+\; 60 \times 1.5 \;+\; 20 \times 0.5 \;=\; 30 + 90 + 10 \;=\; 130 \text{ miles}.$$

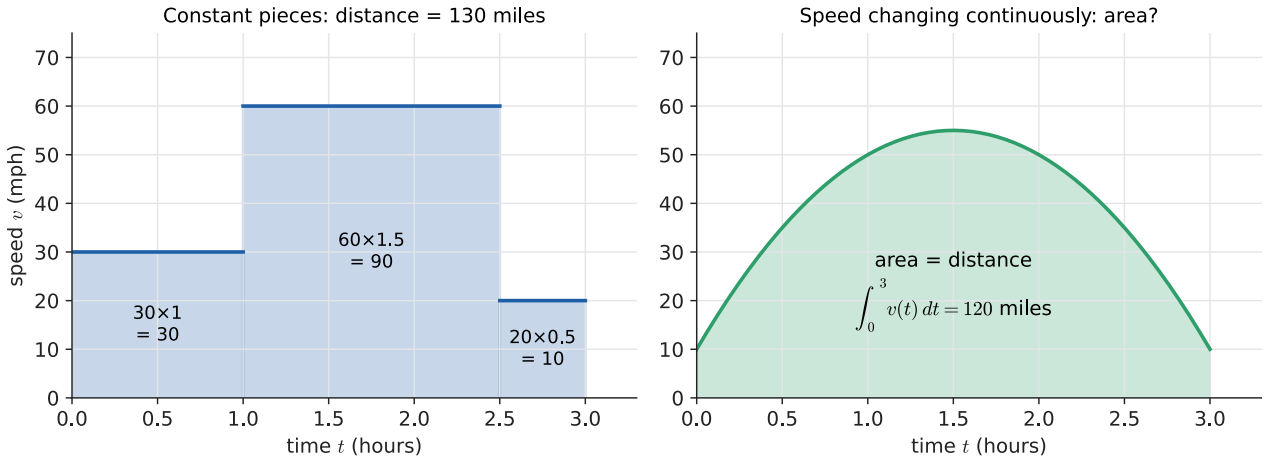


Figure 35. Distance is area under the speed graph. With piecewise-constant speed (left) the area is a sum of rectangles. With smoothly changing speed (right) it is not obvious how to compute it; the answer, 120 miles, will come from the Fundamental Theorem.

19.1.2. Changing speed is the real problem

Real cars do not change speed in jumps. Suppose the speed in mph after t hours is

$$v(t) = 20t(3 - t) + 10, \qquad 0 \le t \le 3.$$

It starts at 10 mph, peaks at 55 mph at $t = 1.5$, and falls back to 10 mph. How far did the car go?

There is no single speed to multiply by 3 hours. But the rectangle picture suggests what to do. On a short enough time interval the speed hardly changes, so "speed times time" is nearly right for that sliver. Chop the journey into many slivers, treat the speed as constant on each, add up the little distances, and make the slivers thinner and thinner. That is the whole idea of the integral, and it is the same limiting move you met with derivatives, applied to a sum instead of a ratio.

Notice what has happened. We started with a physical question (distance from speed) and ended with a geometric one (the area under a curve). These are the same problem, and the integral is the tool that answers both. By the end of the section on the Fundamental Theorem we will find that this car covered exactly 120 miles.

Notice too how this reverses the previous chapter. There, position $s(t)$ was given and we differentiated to get speed $v(t) = s'(t)$. Here the speed is given and we want the change in position. Integration will turn out to be the inverse of differentiation, which is why it is sometimes called **antidifferentiation**.



Accumulation

An **accumulated quantity** is a total built up from a rate over time (or over any other variable). Distance accumulates from speed, water in a tank accumulates from flow rate, money accumulates from income per month. If the rate is constant, total = rate × duration. If the rate varies, total = area under the rate graph. The **integral** is the tool for computing that area.

19.2. Riemann sums: rectangles under a curve

19.2.1. The idea

Take the simplest curved example: the area under $y = x^2$ between $x = 0$ and $x = 2$. No school formula gives this region's area: it is not a triangle, rectangle or circle segment. So we approximate it.

Cut the interval $[0, 2]$ into n equal pieces. Above each piece, build a rectangle whose height is the curve's height **somewhere** in that piece. Add the rectangle areas. The choice of "somewhere" gives three standard versions:

- **Left sum:** use the height at the left end of each piece.
- **Right sum:** use the height at the right end.
- **Midpoint sum:** use the height at the middle.

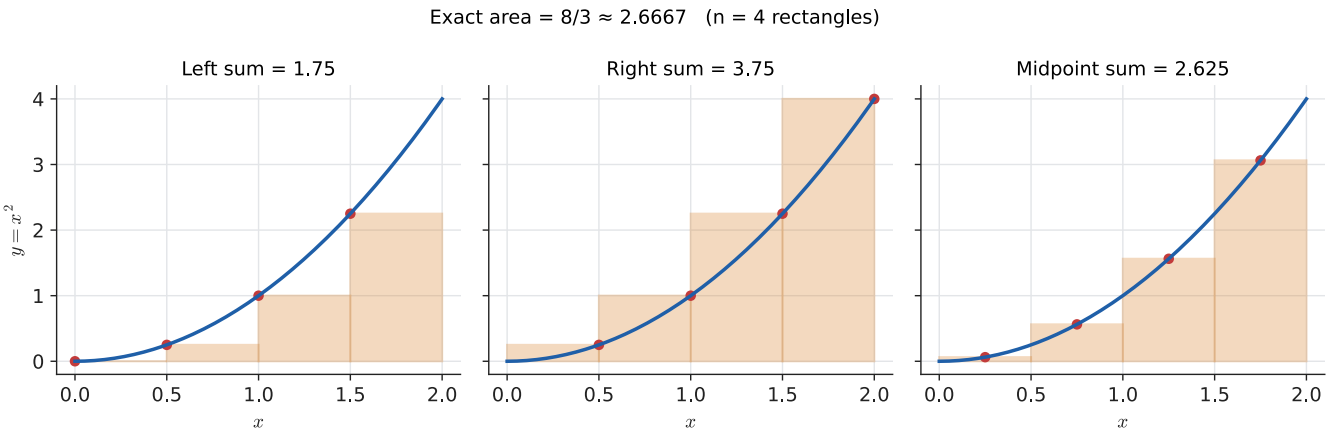


Figure 36. Left, right and midpoint sums for $y = x^2$ on $[0, 2]$ with four rectangles. Red dots mark where each height is measured.

Because x^2 is increasing on this interval, the left sum leaves gaps under the curve (an underestimate) and the right sum overshoots (an overestimate). The midpoint sum overshoots on half of each piece and undershoots on the other half, so the errors largely cancel.

19.2.2. Vocabulary and notation

Partition, subinterval, width

A **partition** of $[a, b]$ splits it into smaller **subintervals** using points $a = x_0 < x_1 < \cdots < x_n = b$. With equal widths, each subinterval has width

$$\Delta x = \frac{b - a}{n},$$

and the partition points are $x_i = a + i\,\Delta x$ for $i = 0, 1, \ldots, n$. The symbol Δ ("delta") means "a change in", so Δx is read "delta x": a small step in x.

Sample point

A **sample point** x_i^* (read "x i star") is the point inside the i-th subinterval where you measure the height $f(x_i^*)$. Left, right and midpoint sums are three particular choices of sample point.

Sigma notation

The capital Greek letter Σ (sigma, for "sum") abbreviates a long addition:

$$\sum_{i=1}^{n} f(x_i^*)\,\Delta x \;=\; f(x_1^*)\,\Delta x + f(x_2^*)\,\Delta x + \cdots + f(x_n^*)\,\Delta x.$$

Piece by piece: i is the **index** (a loop counter); $i = 1$ underneath is where it starts; n on top is where it stops (inclusive); the expression to the right is the **term** evaluated for each i and then added. In code it is `sum(f(x_star[i]) * dx for i in range(1, n + 1))`.

Riemann sum

A **Riemann sum** for f on $[a, b]$ is any sum of the form $\sum_{i=1}^{n} f(x_i^*)\,\Delta x$: heights times widths, added. Each term is the (signed) area of one thin rectangle. Named after Bernhard Riemann (1826–1866), who made the definition precise.

19.2.3. Worked example: three sums by hand

Approximate the area under $f(x) = x^2$ on $[0, 2]$ using $n = 4$. *Try it:* Draw the three sums ; Watch the rectangles .

Step 1: the width. $\Delta x = (2 - 0)/4 = 0.5$.

Step 2: the partition points. $x_0 = 0,\ x_1 = 0.5,\ x_2 = 1,\ x_3 = 1.5,\ x_4 = 2.$

Step 3: left sum. Heights at $0, 0.5, 1, 1.5$:

$$\begin{aligned} L_4 &= 0.5\,\big(f(0) + f(0.5) + f(1) + f(1.5)\big) \\ &= 0.5\,(0 + 0.25 + 1 + 2.25) \\ &= 0.5 \times 3.5 = 1.75. \end{aligned}$$

Step 4: right sum. Heights at $0.5, 1, 1.5, 2$:

$$\begin{aligned} R_4 &= 0.5\,\big(f(0.5) + f(1) + f(1.5) + f(2)\big) \\ &= 0.5\,(0.25 + 1 + 2.25 + 4) \\ &= 0.5 \times 7.5 = 3.75. \end{aligned}$$

Step 5: midpoint sum. Midpoints are $0.25, 0.75, 1.25, 1.75$:

$$\begin{aligned} M_4 &= 0.5\,\big(f(0.25) + f(0.75) + f(1.25) + f(1.75)\big) \\ &= 0.5\,(0.0625 + 0.5625 + 1.5625 + 3.0625) \\ &= 0.5 \times 5.25 = 2.625. \end{aligned}$$

The true area, as we will soon prove, is $8/3 \approx 2.6667$. The midpoint sum is already within 0.042 of it with only four rectangles; the left and right sums are off by about 0.92 and 1.08.

Example 13. Procedure — Riemann sum

Input: a function f, an interval $[a, b]$, a number of subintervals n, and a rule (left, right or midpoint).

Steps:

1. Compute the width $\Delta x = (b - a)/n$.
2. For each $i = 0, 1, \ldots, n - 1$, choose the sample point: $a + i\Delta x$ (left), $a + (i + 1)\Delta x$ (right), or $a + (i + \tfrac{1}{2})\Delta x$ (midpoint).
3. Evaluate f at each sample point.
4. Add the heights and multiply the total by Δx.

Output: an approximation to the area under f on $[a, b]$. It improves as n grows.

PYTHON

```python
def riemann_sum(f, a, b, n, rule="mid"):
    """Approximate the integral of f on [a, b] with n rectangles."""
    dx = (b - a) / n
    offset = {"left": 0.0, "right": 1.0, "mid": 0.5}[rule]
    total = 0.0
    for i in range(n):
        x_sample = a + (i + offset) * dx   # where we measure the height
        total += f(x_sample) * dx          # height times width
    return total

f = lambda x: x**2
for n in [4, 16, 64, 256, 1024]:
    print(f"n={n:5d}  left={riemann_sum(f,0,2,n,'left'):.6f}  "
          f"right={riemann_sum(f,0,2,n,'right'):.6f}  "
          f"mid={riemann_sum(f,0,2,n,'mid'):.6f}")
```

```
n=    4  left=1.750000  right=3.750000  mid=2.625000
n=   16  left=2.421875  right=2.921875  mid=2.664062
n=   64  left=2.604492  right=2.729492  mid=2.666504
n=  256  left=2.651062  right=2.682312  mid=2.666656
n= 1024  left=2.662762  right=2.670574  mid=2.666666
```

All three columns close in on $\mathbf{2.6666\ldots}$. Look at the rates: each time n is multiplied by 4, the left and right errors shrink by roughly a factor of 4, while the midpoint error shrinks by roughly 16. The midpoint rule is not just a little better; it improves **faster**. We will see why in Section 19.12.

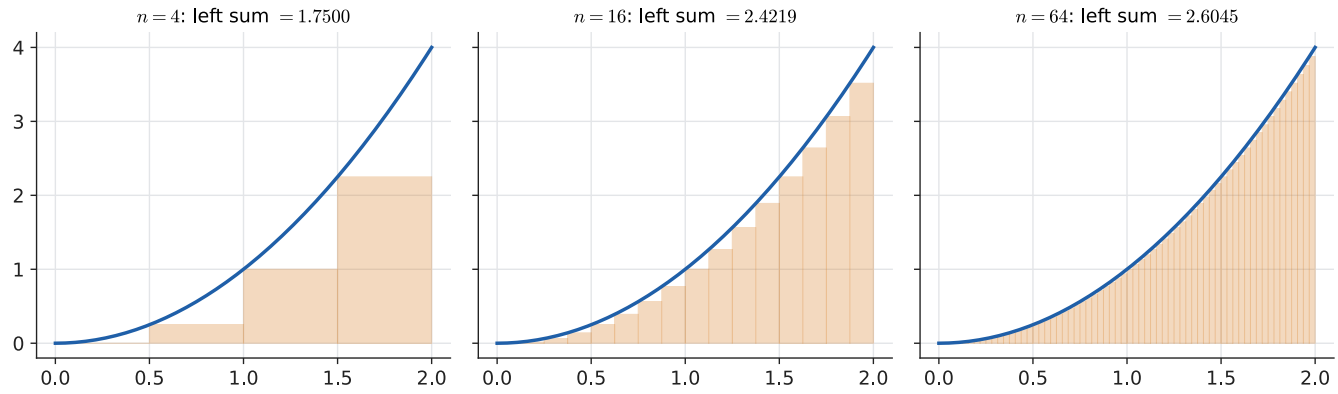


Figure 37. Thinner rectangles hug the curve more closely. Left sums with 4, 16 and 64 rectangles.

19.2.4. Why it works: the exact limit by algebra

For this one function we can compute the sum exactly for **every** n and watch the limit happen. Use right endpoints: $\Delta x = 2/n$ and $x_i = 2i/n$, so $f(x_i) = 4i^2/n^2$.

$$
\begin{aligned}
R_n &= \sum_{i=1}^{n} f(x_i)\,\Delta x = \sum_{i=1}^{n} \frac{4i^2}{n^2}\cdot\frac{2}{n} \\
&= \frac{8}{n^3}\sum_{i=1}^{n} i^2 && \text{(constants come out of the sum)} \\
&= \frac{8}{n^3}\cdot\frac{n(n+1)(2n+1)}{6} && \text{(sum-of-squares formula)} \\
&= \frac{4}{3}\cdot\frac{(n+1)(2n+1)}{n^2} \\
&= \frac{4}{3}\left(1+\frac{1}{n}\right)\left(2+\frac{1}{n}\right).
\end{aligned}
$$

Check with $n = 4$: $\frac{4}{3} \times 1.25 \times 2.25 = 3.75$, matching the hand calculation. As $n \to \infty$, both $1/n$ terms vanish and

$$\lim_{n\to\infty} R_n = \frac{4}{3} \times 1 \times 2 = \frac{8}{3}.$$

That is a genuine derivation of the area under a parabola. It also shows why we need something better: this needed a special formula for $\sum i^2$, and a function like $\sin x$ or e^{-x^2} has no such convenient formula. The Fundamental Theorem will remove the need for these sums entirely.



Which sample point?

"Left sums underestimate" is only true for **increasing** functions. For a decreasing function the left sum overestimates. For a function that rises and falls, neither rule is reliably above or below. Reason from the picture, not from a memorised slogan.

19.2.5. Where you'd use this

Riemann sums are not just a stepping stone to the integral; they are how totals are computed from measurements in practice.

- **Everyday: your electricity bill.** A UK smart meter records your average power every half-hour. If four consecutive half-hours average 0.4, 1.2, 3.0 and 0.8 kW (the 3.0 is the oven), the energy used is the Riemann sum $(0.4 + 1.2 + 3.0 + 0.8) \times 0.5 = 2.7$ kWh, about 68p at 25p per kWh. Your bill is a sum of 48 such rectangles for every day of the month.
- **Surveying and civil engineering: river flow.** Hydrologists measure a river's depth every 2 m across a 10 m channel: 0, 1.2, 2.0, 1.8, 0.9, 0 m. Treating each strip as a trapezoid gives a cross-section of $2 \times (1.2 + 2.0 + 1.8 + 0.9) = 11.8$ m², and at an average current of 0.5 m/s the river carries about 5.9 m³ of water per second. Flood forecasts start with sums like this.
- **Computing.** Any monitoring dashboard that turns a sampled rate (requests per second, watts, bytes per second) into a total is computing a Riemann sum, as the sidebar below explains.

Computer-science connection: a Riemann sum is a fold

A Riemann sum is a loop with an accumulator: `total += f(x) * dx`. In functional terms it is a `map` (evaluate f at the sample points) followed by a `reduce` that adds. That structure is why integration parallelises so well: split the interval across threads, sum each part, then add the partial sums. GPU reductions and MapReduce jobs computing totals are doing Riemann sums in spirit.



Area by counting

Archimedes (c. 250 BCE) found the area under a parabolic arc by filling it with an infinite sequence of triangles whose areas form a geometric series, the "method of exhaustion". He got the right answer nearly two thousand years before calculus, using essentially the same idea as a Riemann sum: approximate with simple shapes, then argue the error can be made as small as you like.



Rectangles converging

Pick a function and an interval. Drag a slider for n from 1 to 200 and switch between left, right and midpoint rules. Watch the rectangles and a live readout of the sum and its error against the exact area. Try a decreasing function and a function that rises and falls to test the "underestimate" slogan.

Try it: Open the rectangles lab · Left sums of a decreasing f · A function that rises and falls · Riemann rectangles, slide n

19.3. The definite integral

19.3.1. The definition

The **definite integral** of f from a to b is the limit of Riemann sums as the rectangles become infinitely thin:

$$\int_a^b f(x)\,dx \;=\; \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i^*)\,\Delta x.$$

For continuous f (and many others) the limit exists and is the same whichever sample points you use, which is why left, right and midpoint sums all converged to the same 8/3.

19.3.2. Reading the notation piece by piece

The notation is a frozen picture of the Riemann sum.

Symbol	Meaning
$\int$	An elongated S, for *summa* (sum). It replaces Σ once the sum has gone to the limit.
a, b	The **lower** and **upper limits of integration**: where the interval starts and ends. (Not "limits" in the $\lim$ sense; just endpoints.)
$f(x)$	The **integrand**: the function whose area we accumulate. It plays the role of the rectangle heights $f(x_i^*)$.
dx	The infinitely thin width that replaces Δx. It also names the **variable of integration**: the variable that runs from a to b.

So $\int_0^2 x^2\,dx$ reads "the integral from 0 to 2 of x^2 d-x" and means "add up x^2 times a sliver of width, for all x from 0 to 2".



Definite integral

The **definite integral** $\int_a^b f(x)\,dx$ is a **number**: the limit of Riemann sums of f over $[a, b]$. Geometrically it is the **signed area** between the graph of f and the x-axis from a to b. A function for which this limit exists is called **integrable** on $[a, b]$.



Dummy variable

The variable of integration is a **dummy variable**: its name does not matter. $\int_0^2 x^2\,dx$, $\int_0^2 t^2\,dt$ and $\int_0^2 u^2\,du$ are all the same number, $8/3$. It is exactly like a loop variable or a bound variable in a lambda: `sum(i*i for i in r)` and `sum(k*k for k in r)` are the same.

19.3.3. Signed area

If $f(x) < 0$, the "rectangle height" $f(x_i^*)$ is negative, so that rectangle contributes **negative** area. The definite integral therefore counts area above the axis as positive and area below as negative. For example, $\sin x$ is positive on $(0, \pi)$ and negative on $(\pi, 2\pi)$ with identical shapes, so

$$\int_0^{2\pi} \sin x\,dx = 0.$$

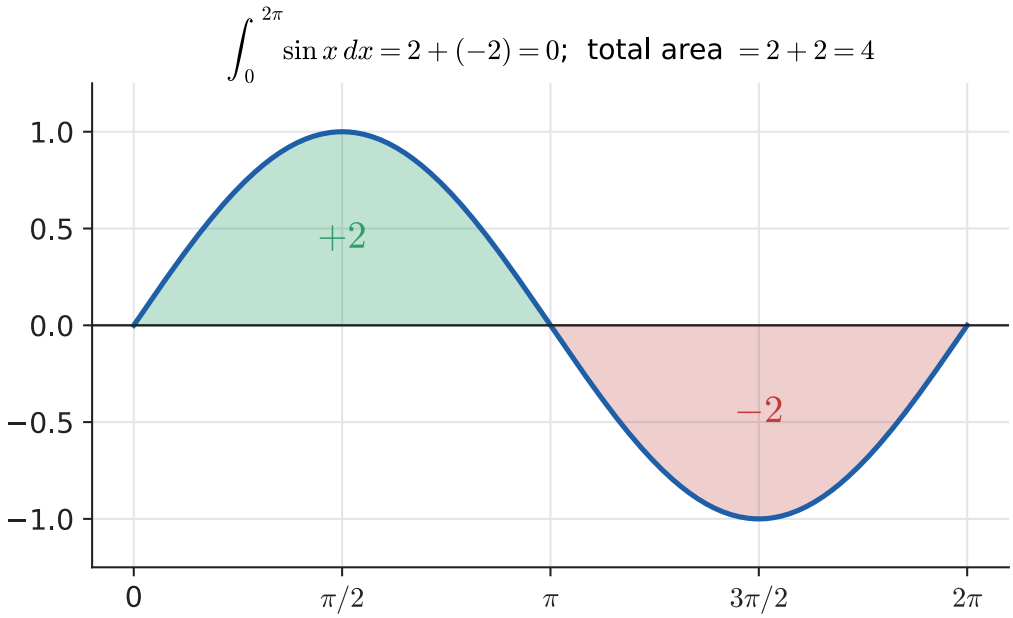


Figure 38. Signed area: the arch above the axis contributes $+2$, the identical arch below contributes -2, so the integral over $[0, 2\pi]$ is 0 even though the total shaded area is 4.

This is the right behaviour for many applications: if $v(t)$ is a **velocity** (which can be negative when you reverse), $\int v\,dt$ is your **net displacement**, and driving 5 miles forward then 5 back gives 0.



Signed area is not total area

If a question asks for the **area** of a region (a geometric quantity, always positive) or the **total distance** travelled, you must split the interval where f changes sign and add the absolute values of the pieces. $\int_0^{2\pi} \sin x\,dx = 0$, but the total area between the sine curve and the axis over that interval is $2 + 2 = 4$.

19.3.4. Properties

These follow directly from the rectangle picture, and you will use them constantly.

$$\int_a^b \big(f(x) + g(x)\big)\,dx = \int_a^b f(x)\,dx + \int_a^b g(x)\,dx \qquad \text{(add heights, add areas)}$$

$$\int_a^b c\,f(x)\,dx = c\int_a^b f(x)\,dx \qquad \text{(scale heights, scale area)}$$

$$\int_a^c f(x)\,dx = \int_a^b f(x)\,dx + \int_b^c f(x)\,dx \qquad \text{(areas of adjacent pieces add)}$$

$$\int_a^a f(x)\,dx = 0 \qquad \text{(zero width)}$$

$$\int_b^a f(x)\,dx = -\int_a^b f(x)\,dx \qquad \text{(running backwards flips the sign of } \Delta x\text{)}$$

The first two together are called **linearity**. The last one is a definition that keeps the third rule true in all orders.



Why the long S, and why "dx"?

Gottfried Leibniz introduced $\int$ in a manuscript of 29 October 1675, writing it as a stretched letter long-s, the way "s" was then often written in the middle of words. He thought of dx as an "infinitely small" difference and of $\int f\,dx$ as a literal sum of infinitely many infinitely thin rectangles. Mathematicians spent the next two centuries making that idea rigorous (Cauchy, then Riemann, then Lebesgue), but Leibniz's notation was so well chosen that it survived unchanged and still does much of the thinking for you, as you'll see with substitution.



Integrable or not?

Every continuous function on a closed interval is integrable, and so is any function with finitely many jumps. A function that is **not** Riemann-integrable is the Dirichlet function: 1 on rational numbers, 0 on irrationals. Every subinterval contains both kinds of number, so you can make any Riemann sum equal 0 or $b - a$ just by choosing the sample points. Henri Lebesgue's 1902 theory of integration handles it (the answer is 0) and underpins modern probability.

US curriculum

Riemann sums and the definite integral as a limit appear in **AP Calculus AB** (and the first semester of college Calculus I or II). Left/right/midpoint and trapezoidal sums are standard AP exam items, often from a table of data rather than a formula. Sigma notation is first seen in **Algebra II** or **Precalculus**.

19.4. The Fundamental Theorem of Calculus

19.4.1. The problem

Computing $\int_0^2 x^2\,dx$ from the limit took a special summation formula and some algebra. For $\int_0^\pi \sin x\,dx$ the same route is miserable. We need a shortcut, and there is a spectacular one.

19.4.2. Intuition: the rate at which area grows

Fix a starting point a and let the right-hand end vary. Define the **accumulation function**

$$A(x) = \int_a^x f(t)\,dt,$$

the area under f from a up to x. (We use t inside so that x can be the moving endpoint; the dummy variable's name does not matter.)

Ask a derivative question about it: **how fast does A grow as x moves right?** Picture yourself painting the region under the curve with a roller moving left to right. The rate at which paint goes on is the height of the strip under the roller right now. Tall curve, area grows quickly; curve near zero, area barely grows; curve negative, the signed area shrinks.

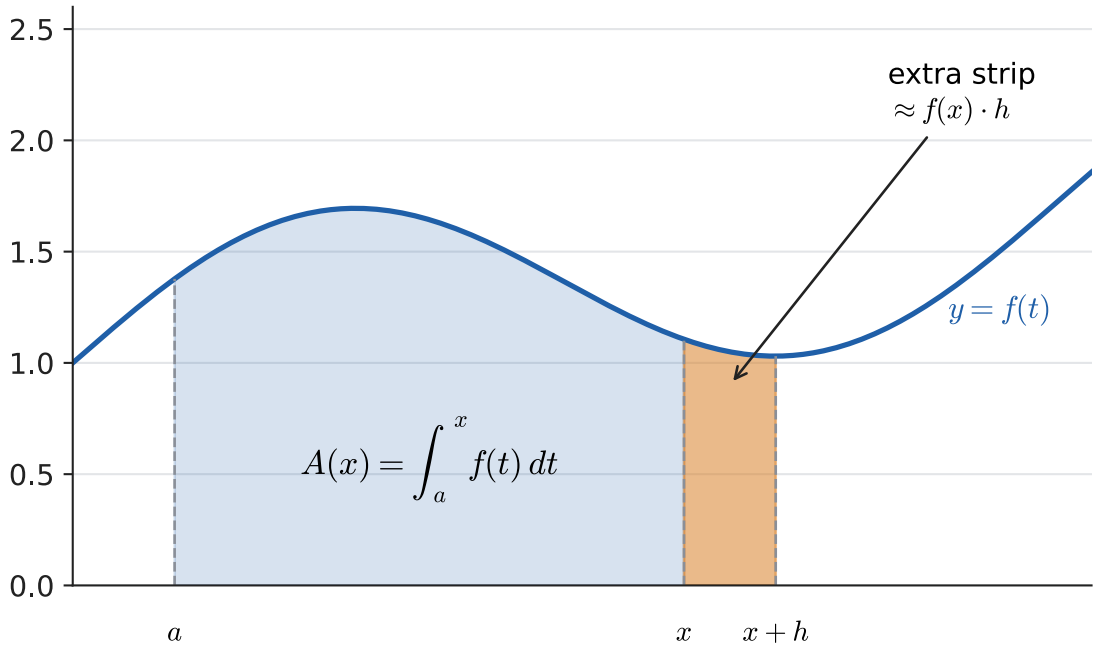


Figure 39. Nudging the endpoint from x to $x + h$ adds a thin strip whose area is about $f(x) \cdot h$.

19.4.3. Why it works

Move the endpoint from x to $x + h$. The extra area is a thin strip of width h and height **about** $f(x)$:

$$A(x + h) - A(x) = \int_x^{x+h} f(t)\,dt \approx f(x) \cdot h.$$

Divide by h:

$$\frac{A(x + h) - A(x)}{h} \approx f(x).$$

The left side is the difference quotient from Chapter 17. As $h \to 0$ the strip gets thinner, the top of the strip flattens out (because f is continuous, its height across the strip approaches $f(x)$), and the approximation becomes exact. So

$$A'(x) = f(x).$$

To make the "approximately" honest: on the strip, f lies between its minimum m_h and maximum M_h, so the strip's area lies between $m_h h$ and $M_h h$, and the difference quotient lies between m_h and M_h. Continuity forces both to approach $f(x)$ as $h \to 0$, so the quotient is squeezed to $f(x)$.



Fundamental Theorem of Calculus, Part 1

If f is continuous on $[a, b]$ and $A(x) = \int_a^x f(t)\,dt$, then A is differentiable and

$$\frac{d}{dx} \int_a^x f(t)\,dt = f(x).$$

In words: **the rate of change of accumulated area is the current height**. Differentiation undoes integration.

19.4.4. From Part 1 to a computing shortcut

Part 1 says A is a function whose derivative is f. Any such function is called an **antiderivative** of f.



Antiderivative

An **antiderivative** of f is a function F with $F'(x) = f(x)$. For example, $F(x) = x^3/3$ is an antiderivative of $f(x) = x^2$, because $\frac{d}{dx}(x^3/3) = x^2$. So is $x^3/3 + 7$: antiderivatives are only determined up to an added constant.

Suppose you have found **some** antiderivative F of f by any means, perhaps by recognising a derivative backwards. Both F and A have derivative f, so their difference has derivative zero, and a function with zero derivative everywhere on an interval is constant (its graph has slope 0 everywhere, so it is flat). Hence

$$A(x) = F(x) + C \quad \text{for some constant } C.$$

Find C by putting $x = a$: the area from a to a is zero, so $0 = A(a) = F(a) + C$, giving $C = -F(a)$. Therefore $A(x) = F(x) - F(a)$, and at $x = b$:



Fundamental Theorem of Calculus, Part 2 (the evaluation theorem)

If f is continuous on $[a, b]$ and F is any antiderivative of f, then

$$\int_a^b f(x)\,dx = F(b) - F(a) = \Big[F(x)\Big]_a^b.$$

The bracket notation $\big[F(x)\big]_a^b$ (also written $F(x)\big|_a^b$) means "evaluate at the top limit, minus evaluate at the bottom limit".

This is astonishing when you first take it in. An infinite limit of sums (an area) is computed by evaluating some other function at **two points**. All the detail of the curve in between is already encoded in F.

19.4.5. Worked examples

Example 1: the parabola again. An antiderivative of x^2 is $x^3/3$. *Try it:* Integrate x² from 0 to 2 .

$$\int_0^2 x^2\,dx = \Big[\frac{x^3}{3}\Big]_0^2 = \frac{2^3}{3} - \frac{0^3}{3} = \frac{8}{3}.$$

One line, versus the page of summation algebra earlier, and the same answer.

Example 2: one arch of the sine curve. Since $\frac{d}{dx}(-\cos x) = \sin x$, an antiderivative of $\sin x$ is $-\cos x$. *Try it:* Paint one arch of sin .

$$\begin{aligned}\int_0^{\pi} \sin x\,dx &= \Big[-\cos x\Big]_0^{\pi} \\ &= (-\cos\pi) - (-\cos 0) \\ &= (-(-1)) - (-1) \\ &= 1 + 1 = 2.\end{aligned}$$

The area under one arch of the sine wave is exactly 2. Nothing about the curve suggested a whole number; the theorem delivers it effortlessly.

Example 3: the car. Back to $v(t) = 20t(3 - t) + 10 = -20t^2 + 60t + 10$. Find an antiderivative term by term (each term: raise the power by one, divide by the new power, which we justify properly in the next section): *Try it:* The car: 120 miles .

$$S(t) = -\frac{20t^3}{3} + 30t^2 + 10t.$$

Check by differentiating: $S'(t) = -20t^2 + 60t + 10 = v(t)$. Good. Now evaluate:

$$\begin{aligned}\int_0^3 v(t)\,dt &= S(3) - S(0) \\ &= \left(-\frac{20 \cdot 27}{3} + 30 \cdot 9 + 10 \cdot 3\right) - 0 \\ &= -180 + 270 + 30 \\ &= 120 \text{ miles}.\end{aligned}$$

The opening question is answered: the car travelled 120 miles. And notice what $S(t)$ **is**: a position function, the distance travelled by time t. The FTC is the formal statement that "integrating speed gives distance" and "differentiating distance gives speed" are inverse operations.



The car's speed is not 40 mph, but its average is

120 miles in 3 hours is an **average** speed of 40 mph. That number has a precise integral meaning (the average value of a function, Section 19.10), and the **Mean Value Theorem for integrals** guarantees the car was going exactly 40 mph at some instant. Here it happens twice, where $-20t^2 + 60t + 10 = 40$, at $t = 1.5 \pm \sqrt{0.75}$, about $t \approx 0.63$ and $t \approx 2.37$ hours.

19.4.6. Part 1 in action

Part 1 is not only theory. It lets you differentiate functions **defined** by integrals, even when the integral has no formula:

$$\frac{d}{dx}\int_0^{x} e^{-t^2}\,dt = e^{-x^2}.$$

If the upper limit is a function of x, combine with the chain rule. With $G(u) = \int_0^u \cos(t^2)\,dt$ and upper limit $u = x^3$:

$$\frac{d}{dx}\int_0^{x^3}\cos(t^2)\,dt = G'(x^3)\cdot 3x^2 = \cos\left(x^6\right)\cdot 3x^2.$$

19.4.7. Where you'd use this

- **Navigation: why phones still need GPS.** A phone or drone can track its motion by integrating its accelerometer: velocity is the integral of acceleration, position the integral of velocity (this is **inertial navigation**). But integration accumulates errors too. A tiny constant sensor bias of 0.01 m/s² integrates to a velocity error of $0.01t$ and a position error of $\int_0^t 0.01s\,ds = 0.005t^2$ metres: 18 m after one minute, 1.8 km after ten. That quadratic growth is why inertial systems are constantly corrected by GPS or other fixes.
- **Statistics.** The probability that a normally distributed quantity is below x is an integral of the bell curve with a variable upper limit, exactly the $\int_0^x e^{-t^2}\,dt$ kind of function above. FTC Part 1 says its derivative is the bell curve itself, which is how statistical software moves between a distribution's **cumulative distribution function** and its **density** (Volume 3, Chapter 30).
- **Engineering.** The charge on a capacitor is the integral of the current that has flowed into it; the energy stored in a battery is the integral of charging power. Battery-management chips in phones and electric cars run this integral continuously ("coulomb counting") to show you the percentage remaining.



Rate in, total out

The FTC needs f continuous on the **whole** interval $[a, b]$. Applying $F(b) - F(a)$ across a point where f blows up gives nonsense. A classic trap is $\int_{-1}^{1} \frac{1}{x^2}\,dx$. Blindly, $\left[-1/x\right]_{-1}^{1} = -1 - 1 = -2$, a **negative** area for a function that is positive everywhere. The real story is that the integral is infinite. See Section 19.11.



Who discovered it?

Isaac Barrow, Newton's teacher at Cambridge, proved a geometric version of the FTC in 1670. Newton (from the mid-1660s, in his "method of fluxions") and Leibniz (from 1675) each turned it into a general **calculus**, a mechanical method for computing areas and rates. Their followers spent decades in a bitter priority dispute. Today both are credited with independently inventing calculus, and it is Leibniz's notation that we use.



Painting the area

Show a curve $y = f(t)$ on top and, underneath, the graph of the accumulation function $A(x) = \int_a^x f(t)\,dt$. Drag the endpoint x and watch the shaded area and $A(x)$ update together. Show the tangent slope of A at x next to the height $f(x)$: they always match. Try a curve that dips below the axis and watch A decrease there.

Try it: Open the painting-the-area lab · f and its area function

US curriculum

Both parts of the Fundamental Theorem of Calculus are central to **AP Calculus AB**. Questions of the form "differentiate $\int_a^{g(x)} f(t)\,dt$" and "interpret $\int_a^b v(t)\,dt$ as displacement" appear on almost every AP exam.

19.5. Antiderivatives and the constant of integration

19.5.1. The problem

The FTC turns every definite integral into a search: **find a function whose derivative is the integrand**. So we need a toolkit for running derivative rules in reverse.

19.5.2. Indefinite integrals and $+C$



Indefinite integral

The **indefinite integral** $\int f(x)\,dx$ (no limits) denotes the **family of all antiderivatives** of f:

$$\int f(x)\,dx = F(x) + C, \qquad \text{where } F'(x) = f(x).$$

C is the **constant of integration**: an arbitrary constant, because adding a constant does not change a derivative. A definite integral is a number; an indefinite integral is a family of functions.

Why must $+C$ be there? Because x^2, $x^2 + 1$ and $x^2 - 5$ all have derivative $2x$. Graphically, they are vertical shifts of the same curve and have identical slopes at every x. Knowing only the slope, you cannot know the height. For a car: knowing your speed at every moment tells you how far you went, but not where you **started**. To pin down C you need one extra fact, an **initial condition** such as "at $t = 0$ the car was at mile marker 40".

In a definite integral C always cancels, $(F(b) + C) - (F(a) + C) = F(b) - F(a)$, which is why we drop it there.

19.5.3. Basic rules: derivative rules run backwards

Each entry is a derivative fact read from right to left. You can check any of them by differentiating the right-hand side.

Integrand $f(x)$	Antiderivative $\int f(x)\,dx$	Check: derivative of the answer
x^n $(n \neq -1)$	$\dfrac{x^{n+1}}{n+1} + C$	$\frac{(n+1)x^n}{n+1} = x^n$
$\dfrac{1}{x}$	$\ln\lvert x\rvert + C$	$\frac{1}{x}$ (for $x > 0$ and $x < 0$)
e^x	$e^x + C$	e^x
e^{kx}	$\dfrac{1}{k}e^{kx} + C$	$\frac{1}{k} \cdot ke^{kx}$ (chain rule)
$\sin x$	$-\cos x + C$	$-(-\sin x)$
$\cos x$	$\sin x + C$	$\cos x$
$\sec^2 x$	$\tan x + C$	$\sec^2 x$
$\dfrac{1}{1+x^2}$	$\arctan x + C$	$\frac{1}{1+x^2}$
a^x	$\dfrac{a^x}{\ln a} + C$	$\frac{a^x \ln a}{\ln a}$

And the **linearity** rules carry over from derivatives: $\int (f+g)\,dx = \int f\,dx + \int g\,dx$ and $\int cf\,dx = c\int f\,dx$.

Example 14. Procedure — evaluating a definite integral with the FTC

Input: an integral $\int_a^b f(x)\,dx$ with f continuous on $[a, b]$.

Steps:

1. Rewrite the integrand as a sum of standard forms (expand brackets, write roots as powers, split fractions).
2. Find an antiderivative F term by term using the table and linearity.
3. **Check** by differentiating F: you must get f back.
4. Compute $F(b)$ and $F(a)$ separately, with brackets around each.
5. Subtract: $F(b) - F(a)$.

Output: the exact value of the integral.

19.5.4. Worked examples

Example 1. Evaluate $\int_1^3 (3x^2 - 4x + 5)\,dx$. *Try it:* Evaluate this integral .

$$F(x) = 3\cdot\frac{x^3}{3} - 4\cdot\frac{x^2}{2} + 5x = x^3 - 2x^2 + 5x \qquad \text{(power rule, term by term)}$$
$$F'(x) = 3x^2 - 4x + 5 \;\checkmark \qquad \text{(check)}$$
$$F(3) = 27 - 18 + 15 = 24$$
$$F(1) = 1 - 2 + 5 = 4$$
$$\int_1^3 (3x^2 - 4x + 5)\,dx = 24 - 4 = 20.$$

Example 2. Find $\displaystyle\int \left(2\sqrt{x} - \frac{3}{x} + e^{2x}\right) dx$.

First rewrite $\sqrt{x} = x^{1/2}$. Then

$$\int 2x^{1/2}\,dx = 2\cdot\frac{x^{3/2}}{3/2} = \frac{4}{3}x^{3/2} \qquad \text{(power rule with } n = \tfrac{1}{2}\text{)}$$
$$\int \frac{3}{x}\,dx = 3\ln|x| \qquad \text{(the } n = -1 \text{ case)}$$
$$\int e^{2x}\,dx = \frac{1}{2}e^{2x} \qquad (e^{kx} \text{ rule, } k = 2)$$
$$\int \left(2\sqrt{x} - \frac{3}{x} + e^{2x}\right) dx = \frac{4}{3}x^{3/2} - 3\ln|x| + \frac{1}{2}e^{2x} + C.$$

Check: $\frac{d}{dx}\left(\frac{4}{3}x^{3/2}\right) = \frac{4}{3}\cdot\frac{3}{2}x^{1/2} = 2\sqrt{x}$; the other two terms differentiate back correctly too.

Example 3: using an initial condition. A function satisfies $f'(x) = 6x - 2$ and $f(1) = 5$. Find f.

$$f(x) = \int (6x - 2)\,dx = 3x^2 - 2x + C$$
$$f(1) = 3 - 2 + C = 1 + C = 5$$
$$C = 4$$
$$f(x) = 3x^2 - 2x + 4.$$

19.5.5. Mechanics: where the "suvat" equations come from

If you did UK Mechanics you memorised the constant-acceleration equations. They are just two integrations. Let acceleration be the constant a, with initial velocity u and initial position 0 (the old notation used s for displacement and v for velocity).

Velocity is the antiderivative of acceleration:

$$v(t) = \int a\,dt = at + C_1, \qquad v(0) = u \;\Rightarrow\; C_1 = u, \qquad v = u + at.$$

Displacement is the antiderivative of velocity:

$$s(t) = \int (u + at)\,dt = ut + \tfrac{1}{2}at^2 + C_2, \qquad s(0) = 0 \;\Rightarrow\; s = ut + \tfrac{1}{2}at^2.$$

A ball thrown straight up at $u = 20\ \text{m/s}$ with $a = -9.8\ \text{m/s}^2$ has $v(2) = 20 - 19.6 = 0.4$ m/s and $s(2) = 40 - 19.6 = 20.4$ m. We will reproduce those numbers numerically in Section 19.13. The area under the velocity–time graph is the displacement, exactly as the Mechanics syllabus said, and now you know why.



Common antiderivative mistakes

- **The power rule fails at $n = -1$.** $x^{-1+1}/0$ is meaningless; the antiderivative of $1/x$ is $\ln|x|$.
- **The integral of a product is not the product of integrals.** $\int x \cdot x\,dx = x^3/3$, not $(x^2/2)(x^2/2)$. Products need substitution or integration by parts.
- **The integral of a quotient is not a quotient of integrals.** Split or simplify first: $\int \frac{x^2+1}{x}\,dx = \int (x + \frac{1}{x})\,dx$.
- **Forgetting $+C$** in an indefinite integral loses all but one member of the family, which matters as soon as an initial condition appears.
- **Sign of $\int \sin x\,dx$.** It is $-\cos x$. Check by differentiating whenever unsure.



Differentiation is mechanical, integration is an art

Every combination of elementary functions has an elementary derivative, found mechanically by the rules. Integration is different: many innocent-looking functions have **no** antiderivative expressible with elementary functions. e^{-x^2}, $\sin(x)/x$, $\sqrt{1+x^3}$ and $\cos(x^2)$ are examples. Their integrals still exist as numbers (areas), so we either give them new names (e^{-x^2} leads to the **error function erf**, essential in statistics) or compute them numerically. Liouville in the 1830s proved such results, and the **Risch algorithm** (1969), used in computer algebra systems, decides whether an elementary antiderivative exists.

US curriculum

Antiderivatives, indefinite integrals and initial-value problems are **AP Calculus AB** topics. The motion problems (position, velocity, acceleration) are a staple of the AP exam and correspond to the UK Mechanics "suvat" material.

19.6. Exercises: Riemann sums, the definite integral and the FTC

1. **Recognition.** In $\int_{-1}^{4} (t^3 + 1)\,dt$, name the integrand, the limits of integration and the variable of integration. Is the result a number or a function?
2. **Recognition.** Without calculating, explain why $\int_{-1}^{1} x^3\,dx = 0$.
3. **Application.** For $f(x) = x^3$ on $[0, 2]$ with $n = 4$, compute the left, right and midpoint sums. Compare with the exact value.

4. **Application.** Use geometry (not antiderivatives) to evaluate $\int_{-2}^{2} \sqrt{4 - x^2}\, dx$ and $\int_0^3 (2x - 2)\, dx$.
5. **Application.** Evaluate $\int_0^1 (x^4 - 2x + 3)\, dx$, $\int_1^e \frac{1}{x}\, dx$ and $\int_0^{\pi/2} \cos x\, dx$.
6. **Application.** Find $\frac{d}{dx} \int_0^x \cos(t^2)\, dt$ and $\frac{d}{dx} \int_1^{x^2} \frac{\sin t}{t}\, dt$.
7. **Problem solving.** A particle has acceleration $a(t) = 6t$ m/s², initial velocity 2 m/s and initial position 1 m. Find its position at $t = 2$.

▼ **Answers**

1. Integrand $t^3 + 1$; lower limit -1, upper limit 4; variable t. The result is a number.
2. x^3 is an **odd** function: $(-x)^3 = -x^3$. The area from -1 to 0 is a mirror image (below the axis) of the area from 0 to 1 (above), so the signed areas cancel.
3. $\Delta x = 0.5$. Left: $0.5(0 + 0.125 + 1 + 3.375) = 2.25$. Right: $0.5(0.125 + 1 + 3.375 + 8) = 6.25$. Midpoint: $0.5(0.015625 + 0.421875 + 1.953125 + 5.359375) = 0.5 \times 7.75 = 3.875$. Exact: $\left[x^4/4\right]_0^2 = 4$. The midpoint sum is closest.
4. $y = \sqrt{4 - x^2}$ is the top half of a circle of radius 2, so the area is $\frac{1}{2}\pi \cdot 2^2 = 2\pi$. For $2x - 2$: from 0 to 1 it is below the axis, a triangle of area $\frac{1}{2} \cdot 1 \cdot 2 = 1$, counted as -1; from 1 to 3 it is a triangle of area $\frac{1}{2} \cdot 2 \cdot 4 = 4$. Total signed area $4 - 1 = 3$. (Check: $\left[x^2 - 2x\right]_0^3 = 9 - 6 = 3$.)
5. $\left[x^5/5 - x^2 + 3x\right]_0^1 = 0.2 - 1 + 3 = 2.2$. $\left[\ln x\right]_1^e = 1 - 0 = 1$. $\left[\sin x\right]_0^{\pi/2} = 1 - 0 = 1$.
6. By FTC Part 1, the first is $\cos(x^2)$. For the second, with upper limit $u = x^2$: $\frac{\sin(x^2)}{x^2} \cdot 2x = \frac{2\sin(x^2)}{x}$.
7. $v(t) = 3t^2 + C_1$, $v(0) = 2$ gives $v = 3t^2 + 2$. $s(t) = t^3 + 2t + C_2$, $s(0) = 1$ gives $s = t^3 + 2t + 1$. So $s(2) = 8 + 4 + 1 = 13$ m.

19.7. Substitution: the chain rule in reverse

19.7.1. The problem

What is $\int 2x \cos(x^2)\, dx$? It is not in the table, and it is a product, so linearity does not help. But stare at it: x^2 sits inside the cosine, and $2x$, the derivative of x^2, sits outside. That pattern is the fingerprint of the chain rule.

19.7.2. Why it works

The chain rule (Chapter 17) says

$$\frac{d}{dx} F\big(g(x)\big) = F'\big(g(x)\big)\, g'(x).$$

Read it backwards: if an integrand has the form $f(g(x))\, g'(x)$ and F is an antiderivative of f, then

$$\int f\big(g(x)\big)\, g'(x)\, dx = F\big(g(x)\big) + C.$$

Leibniz notation makes this almost automatic. Name the inner function $u = g(x)$. Then $\frac{du}{dx} = g'(x)$, which we write as the **differential** statement $du = g'(x)\,dx$. The integrand's $g'(x)\,dx$ becomes du and the inner function becomes u:

$$\int f(\underbrace{g(x)}_{u})\,\underbrace{g'(x)\,dx}_{du} = \int f(u)\,du.$$



u-substitution

Substitution (or **u-substitution**, or **change of variables**) rewrites an integral in terms of a new variable $u = g(x)$, replacing $g'(x)\,dx$ by du. It is the chain rule run backwards. It works when the integrand contains an inner function **and** (up to a constant factor) that inner function's derivative.

Example 15. Procedure — u-substitution

Input: an integral $\int h(x)\,dx$ (or $\int_a^b h(x)\,dx$).

Steps:

1. Spot an inner function $u = g(x)$ whose derivative $g'(x)$ also appears as a factor (a constant multiple is fine).
2. Compute $du = g'(x)\,dx$ and solve for whatever piece of the integrand you need, e.g. $x\,dx = \frac{1}{2}du$.
3. Rewrite the **entire** integrand in terms of u and du. No x may remain; if it does, choose a different u or a different method.
4. For a definite integral, convert the limits too: $x = a \mapsto u = g(a)$ and $x = b \mapsto u = g(b)$.
5. Integrate in u.
6. Indefinite: substitute $u = g(x)$ back and add C. Definite: evaluate at the new limits (no back-substitution needed).

Output: the antiderivative or the value of the integral. Check an antiderivative by differentiating.

19.7.3. Worked examples

Example 1. $\int 2x\cos(x^2)\,dx$. *Try it:* See u = x² stretch strips ; Integrate 2x cos(x²) .

$$\begin{aligned} & u = x^2, \qquad du = 2x\,dx && \text{(choose the inner function)} \\ \int 2x\cos(x^2)\,dx &= \int \cos u\,du && (2x\,dx \text{ becomes } du) \\ &= \sin u + C \\ &= \sin(x^2) + C. && \text{(back-substitute)} \end{aligned}$$

Check: $\frac{d}{dx}\sin(x^2) = \cos(x^2)\cdot 2x$. Correct.

Example 2: a missing constant. $\int \frac{x}{x^2+1}\,dx$. *Try it:* Integrate x/(x² + 1) .

$$\begin{aligned} u &= x^2+1, \qquad du = 2x\,dx \;\Rightarrow\; x\,dx = \tfrac{1}{2}\,du \\ \int \frac{x}{x^2+1}\,dx &= \int \frac{1}{u}\cdot\tfrac{1}{2}\,du = \tfrac{1}{2}\ln|u| + C \\ &= \tfrac{1}{2}\ln(x^2+1) + C. \qquad (x^2+1>0, \text{ so no absolute value needed}) \end{aligned}$$

Example 3: a definite integral with changed limits. $\int_0^1 x(x^2+1)^3\,dx$. *Try it:* Substitution strips for Example 3 ; Evaluate this integral .

$$\begin{aligned} u &= x^2+1, \qquad du = 2x\,dx \;\Rightarrow\; x\,dx = \tfrac{1}{2}\,du \\ x &= 0 \mapsto u = 1, \qquad x = 1 \mapsto u = 2 \qquad (\text{convert the limits}) \\ \int_0^1 x(x^2+1)^3\,dx &= \int_1^2 u^3\cdot\tfrac{1}{2}\,du \\ &= \tfrac{1}{2}\left[\frac{u^4}{4}\right]_1^2 \\ &= \tfrac{1}{2}\left(\frac{16}{4} - \frac{1}{4}\right) = \tfrac{1}{2}\cdot\frac{15}{4} = \frac{15}{8}. \end{aligned}$$

Alternative approach for Example 3. You could also find the indefinite integral $\frac{1}{8}(x^2+1)^4 + C$, back-substitute, and evaluate with the original limits: $\frac{1}{8}(2^4) - \frac{1}{8}(1^4) = \frac{15}{8}$. Same answer. Converting the limits is usually less work and avoids one source of error; back-substituting is handy when you want the antiderivative anyway. A third alternative, expanding $(x^2+1)^3$ into a polynomial, works here but becomes hopeless for $(x^2+1)^{20}$ and impossible for $(x^2+1)^{1/2}$.



Substitution pitfalls

- **Leftover x.** $\int \cos(x^2)\,dx$ looks similar to Example 1, but there is no factor x to absorb into du. You cannot write $dx = du/(2x)$ and then treat x as a constant. In fact $\cos(x^2)$ has no elementary antiderivative at all.
- **Mixing limits.** In a definite integral, either convert the limits to u or back-substitute before evaluating, never half of each. Evaluating $\frac{1}{2}\left[u^4/4\right]$ at x-limits 0 and 1 gives the wrong $\frac{1}{8}$.
- **Forgetting the constant factor** (the $\frac{1}{2}$ in Example 2) is the most common slip.

19.8. Integration by parts: the product rule in reverse

19.8.1. The problem

Substitution handles "function of a function times the inner derivative". It cannot do $\int xe^x\,dx$: there is no inner function whose derivative is lying around. This is a **product of two unrelated kinds of function**, a polynomial and an exponential, and its reverse partner is the product rule.

19.8.2. Why it works

The product rule says $(uv)' = u'v + uv'$. Integrate both sides:

$$uv = \int u'v\,dx + \int uv'\,dx.$$

Rearrange to isolate one of the integrals:

$$\int u\,v'\,dx = uv - \int u'\,v\,dx.$$

In differential shorthand, with $du = u'\,dx$ and $dv = v'\,dx$:

$$\int u\,dv = uv - \int v\,du.$$

The formula trades one integral for another. It helps when the new integral $\int v\,du$ is **easier** than the original. The trick is to choose u as the factor that gets simpler when differentiated ($x \to 1$, $\ln x \to 1/x$) and dv as the factor you can easily integrate without it getting worse (e^x, $\sin x$).



Integration by parts

Integration by parts is the formula $\int u\,dv = uv - \int v\,du$, obtained by integrating the product rule. For definite integrals: $\int_a^b u\,dv = \left[uv\right]_a^b - \int_a^b v\,du$. A rough guide for choosing u is **LIATE**: Logarithmic, Inverse trig, Algebraic (powers of x), Trigonometric, Exponential, taking as u whichever comes first in that list.

Example 16. Procedure — integration by parts

Input: an integral $\int f(x)\,g(x)\,dx$ of a product.

Steps:

1. Choose u (the factor that simplifies when differentiated; use LIATE) and let dv be everything else, including dx.
2. Compute $du = u'\,dx$ by differentiating, and v by integrating dv (no $+C$ needed here).
3. Write $uv - \int v\,du$.
4. Evaluate the new integral. If it is no easier, go back and swap the choices.
5. Add $+C$ (indefinite) or evaluate between limits (definite).

Output: the antiderivative or value. Check by differentiating.

19.8.3. Worked examples

Example 1. $\int xe^x\,dx$. Algebraic beats exponential in LIATE, so $u = x$. *Try it:* Integrate x e^x .

$$
\begin{aligned}
u &= x, & dv &= e^x\,dx \\
du &= dx, & v &= e^x
\end{aligned}
$$

$$
\begin{aligned}
\int x e^x\,dx &= x e^x - \int e^x\,dx && (uv - \textstyle\int v\,du) \\
&= x e^x - e^x + C \\
&= e^x(x-1) + C.
\end{aligned}
$$

Check: $\frac{d}{dx}\left(e^x(x-1)\right) = e^x(x-1) + e^x \cdot 1 = x e^x$. Correct.

What if we had chosen the other way, $u = e^x$, $dv = x\,dx$? Then $v = x^2/2$ and the new integral is $\int \frac{x^2}{2} e^x\,dx$, which is **worse**. That is the signal to swap.

Example 2: the "invisible" second factor. $\int \ln x\,dx$. There seems to be only one factor, but write it as $\ln x \cdot 1$. *Try it:* Integrate ln x .

$$
\begin{aligned}
u &= \ln x, & dv &= 1\,dx \\
du &= \tfrac{1}{x}\,dx, & v &= x
\end{aligned}
$$

$$
\begin{aligned}
\int \ln x\,dx &= x\ln x - \int x \cdot \frac{1}{x}\,dx \\
&= x\ln x - \int 1\,dx \\
&= x\ln x - x + C.
\end{aligned}
$$

Example 3: a definite integral. $\int_0^{\pi} x \sin x\,dx$.

$$
\begin{aligned}
u &= x, & dv &= \sin x\,dx \\
du &= dx, & v &= -\cos x
\end{aligned}
$$

$$
\begin{aligned}
\int_0^{\pi} x\sin x\,dx &= \Big[-x\cos x\Big]_0^{\pi} - \int_0^{\pi} (-\cos x)\,dx \\
&= \big(-\pi\cos\pi - 0\big) + \int_0^{\pi} \cos x\,dx \\
&= \big(-\pi\cdot(-1)\big) + \Big[\sin x\Big]_0^{\pi} \\
&= \pi + (0-0) = \pi.
\end{aligned}
$$

Example 4: going round in a circle on purpose. $I = \int e^x \sin x\,dx$. Parts twice, with u the trig factor both times:

$$
\begin{aligned}
I &= e^x\sin x - \int e^x\cos x\,dx && (u=\sin x,\ dv = e^x dx) \\
&= e^x\sin x - \Big(e^x\cos x - \int e^x(-\sin x)\,dx\Big) && (u=\cos x,\ dv = e^x dx) \\
&= e^x\sin x - e^x\cos x - I.
\end{aligned}
$$

The original integral I has reappeared. Treat it as an unknown and solve: $2I = e^x(\sin x - \cos x)$, so

$$\int e^x \sin x \, dx = \tfrac{1}{2} e^x (\sin x - \cos x) + C.$$

Where you'd use this. Integration by parts is how averages of "x times a density" are computed. If the waiting time for the next request at a server has the exponential density $\lambda e^{-\lambda x}$ (see Section 19.10), the mean wait is $\int_0^\infty x\,\lambda e^{-\lambda x}\,dx$. Parts with $u = x$, $dv = \lambda e^{-\lambda x} dx$ gives $\left[-x e^{-\lambda x}\right]_0^\infty + \int_0^\infty e^{-\lambda x}\,dx = 0 + \frac{1}{\lambda}$: at $\lambda = 0.5$ requests per second, the average wait is 2 seconds. The same calculation gives the mean lifetime of a radioactive nucleus and the expected time to failure of a component in reliability engineering. Substitution, meanwhile, is how the normal distribution is **standardised**: $z = (x - \mu)/\sigma$ turns every bell-curve integral into one with $\mu = 0$, $\sigma = 1$, which is why a single z-table (or one library function) serves them all.

19.8.4. Choosing a method

What you see	Method	Example
A sum of standard forms	Table + linearity	$\int (x^3 - 4\cos x)\,dx$
An inner function and its derivative (up to a constant)	Substitution	$\int x^2 e^{x^3}\,dx$
A product of two different kinds of function, one of which simplifies when differentiated	Parts	$\int x^2 \cos x\,dx$ (parts twice)
A single $\ln$ or inverse-trig function	Parts with $dv = dx$	$\int \arctan x\,dx$
A rational function $p(x)/q(x)$	Algebra first (division, **partial fractions**)	$\int \frac{1}{x^2-1}\,dx$
No elementary antiderivative, or only data points	Numerical integration	$\int_0^1 e^{-x^2}\,dx$

Try it: Partial fractions of 1/(x² − 1) .

In practice, professionals reach for a computer algebra system (SymPy, Mathematica) for hard symbolic integrals, and for numerical methods whenever only a number is needed. The point of learning substitution and parts is not to compete with software; it is to **read** the formulas in papers and textbooks (many derivations are a substitution or parts step in disguise) and to be able to check what software tells you.



Partial fractions, trig substitution and the rest

A traditional calculus course spends weeks on further techniques: partial fractions, trigonometric substitution, reduction formulas, tables of hundreds of integrals. The Victorian Cambridge "Tripos" examinations made integration a competitive sport. Today these are worth recognising, not drilling. Richard Feynman famously preferred another trick, "differentiating under the integral sign", which he learned from an old textbook and which let him solve integrals that stumped his colleagues at Los Alamos.



Watching a substitution

Plot $2x\cos(x^2)$ on $[0, \sqrt{\pi}]$ beside $\cos u$ on $[0, \pi]$, with a slider that moves a thin strip in one plot and the corresponding strip in the other. The strips have different widths and heights but the same area: substitution is a stretching of the horizontal axis that preserves area.

Try it: Open the substitution lab

US curriculum

u-substitution is in **AP Calculus AB**. Integration by parts (with partial fractions and improper integrals) is in **AP Calculus BC** and college Calculus II.

19.9. Exercises: techniques of integration

1. **Recognition.** For each, say which method you would use and why: (a) $\int x^2 e^x\,dx$, (b) $\int x^2 e^{x^3}\,dx$, (c) $\int (e^x + x^2)\,dx$.
2. **Application.** $\int 3x^2(x^3+1)^5\,dx$.
3. **Application.** $\int \cos x\, e^{\sin x}\,dx$.
4. **Application.** $\int_0^1 x e^{x^2}\,dx$.
5. **Application.** $\int x\cos x\,dx$.
6. **Problem solving.** $\int_1^e x\ln x\,dx$.
7. **Problem solving.** $\int x^2 e^x\,dx$ (parts twice).

▼ **Answers**

1. (a) Parts: product of a polynomial and an exponential, with no inner-function pattern; $u = x^2$, twice. (b) Substitution: $u = x^3$ and $du = 3x^2\,dx$ is present up to a factor $\frac{1}{3}$. (c) Table and linearity: $e^x + x^3/3 + C$.
2. $u = x^3 + 1$, $du = 3x^2\,dx$: $\int u^5\,du = u^6/6 + C = (x^3+1)^6/6 + C$.
3. $u = \sin x$, $du = \cos x\,dx$: $\int e^u\,du = e^{\sin x} + C$.

4. $u = x^2$, $x\,dx = \frac{1}{2}du$, limits 0 to 1: $\frac{1}{2}\int_0^1 e^u\,du = \frac{1}{2}(e-1) \approx 0.8591$.
5. $u = x$, $dv = \cos x\,dx$, $v = \sin x$: $x\sin x - \int \sin x\,dx = x\sin x + \cos x + C$.
6. $u = \ln x$ (L before A), $dv = x\,dx$, $du = dx/x$, $v = x^2/2$.
   $\left[\frac{x^2}{2}\ln x\right]_1^e - \int_1^e \frac{x}{2}\,dx = \frac{e^2}{2} - \left[\frac{x^2}{4}\right]_1^e = \frac{e^2}{2} - \frac{e^2}{4} + \frac{1}{4} = \frac{e^2+1}{4} \approx 2.0973$.
7. $u = x^2$, $v = e^x$: $x^2e^x - \int 2xe^x\,dx$. Then by Example 1, $\int 2xe^x\,dx = 2e^x(x-1)$. Total: $e^x(x^2 - 2x + 2) + C$. Check: derivative is $e^x(x^2-2x+2) + e^x(2x-2) = x^2e^x$.

19.10. Applications: slice, approximate, sum, integrate

Almost every application of integration follows one pattern, and it is worth stating explicitly because it is how you **set up** integrals for problems no textbook has seen.

Example 17. Procedure — setting up an integral from a physical problem

Input: a quantity Q that is a total of something varying along a variable x from a to b.

Steps:

1. **Slice:** cut the range $[a, b]$ into thin pieces of width Δx.
2. **Approximate:** on one slice at position x, treat everything as constant and write the slice's contribution as $\Delta Q \approx (\text{something in } x) \times \Delta x$.
3. **Sum:** the total is approximately the Riemann sum of those contributions.
4. **Integrate:** let $\Delta x \to 0$, replacing the sum by $\int_a^b (\text{something in } x)\,dx$.
5. Evaluate (FTC, or numerically).

Output: Q as a definite integral, and its value.

19.10.1. Area between curves

Problem. Find the area enclosed between $y = x$ and $y = x^2$. *Try it:* Area between x and x² .

Slice and approximate. The curves meet where $x = x^2$, that is $x(x-1) = 0$, so at $x = 0$ and $x = 1$. Between them, $x > x^2$ (try $x = 0.5$: $0.5 > 0.25$), so a thin vertical slice at position x has height $\text{top} - \text{bottom} = x - x^2$ and area $(x - x^2)\,\Delta x$.

Integrate.

$$\begin{aligned}\text{Area} &= \int_0^1 (x - x^2)\,dx = \left[\frac{x^2}{2} - \frac{x^3}{3}\right]_0^1 \\ &= \frac{1}{2} - \frac{1}{3} = \frac{1}{6}.\end{aligned}$$

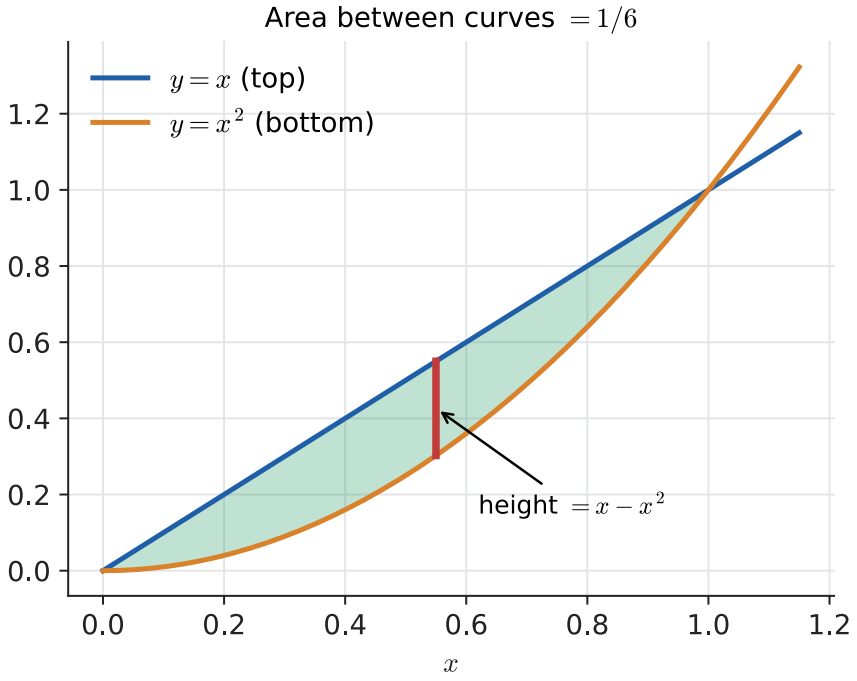


Figure 40. The region between $y = x$ and $y = x^2$. Each vertical slice has height "top minus bottom".

In general, $\text{Area} = \int_a^b \big(\text{top}(x) - \text{bottom}(x)\big)\,dx$. Because we subtract heights, it does not matter whether the curves are above or below the axis. If the curves cross inside the interval, split at the crossings so that "top" really is on top in each piece.

19.10.2. Average value of a function

The average of n numbers is their sum divided by n. For a function, sample it at n evenly spaced points and average:

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i) = \frac{1}{b-a}\sum_{i=1}^{n} f(x_i)\,\Delta x \qquad \left(\text{since } \tfrac{1}{n} = \tfrac{\Delta x}{b-a}\right).$$

The right side is a Riemann sum divided by $b - a$. Letting $n \to \infty$:



Average value

The **average value** of f on $[a, b]$ is

$$\bar{f} = \frac{1}{b-a}\int_a^b f(x)\,dx.$$

Geometrically, it is the height of the rectangle on $[a, b]$ with the same area as the region under f.

Example. The average value of $\sin x$ over one arch $[0, \pi]$ is $\frac{1}{\pi} \cdot 2 = \frac{2}{\pi} \approx 0.6366$. Electrical engineers use exactly this (and the related root-mean-square average) to relate peak and average AC voltages. The car's average speed was $\frac{1}{3} \cdot 120 = 40$ mph.

19.10.3. Volumes of revolution (briefly)

Spin the curve $y = f(x)$, $a \le x \le b$, around the x-axis. The solid it sweeps out can be sliced like a salami into thin discs. The slice at x is a cylinder of radius $f(x)$ and thickness Δx, with volume $\pi f(x)^2\,\Delta x$. Sum and integrate (the **disc method**): *Try it:* Spin a semicircle into a sphere .

$$V = \pi \int_a^b f(x)^2\,dx.$$

Example: a sphere. The upper half of a circle of radius r is $y = \sqrt{r^2 - x^2}$ for $-r \le x \le r$. Spinning it gives a sphere:

$$\begin{aligned} V &= \pi \int_{-r}^{r} (r^2 - x^2)\,dx = \pi\left[r^2 x - \frac{x^3}{3}\right]_{-r}^{r} \\ &= \pi\left[\left(r^3 - \frac{r^3}{3}\right) - \left(-r^3 + \frac{r^3}{3}\right)\right] \\ &= \pi\left(2r^3 - \frac{2r^3}{3}\right) = \frac{4}{3}\pi r^3. \end{aligned}$$

The school formula for the volume of a sphere, which you probably memorised without proof, falls straight out.

19.10.4. Work

In physics, the **work** done by a constant force F moving an object a distance d in the direction of the force is $W = Fd$ (in joules when F is in newtons and d in metres). When the force varies with position, slice:

$$W = \int_a^b F(x)\,dx.$$

Example: stretching a spring. Hooke's law says a spring stretched by x metres pulls back with force $F = kx$. For $k = 200$ N/m, the work to stretch it 0.1 m from rest is

$$W = \int_0^{0.1} 200x\,dx = \Big[100x^2\Big]_0^{0.1} = 100 \times 0.01 = 1\ \text{J}.$$

Example: winding up a hanging chain. A 10 m chain of mass 2 kg/m hangs from a winch. The slice of chain at depth y (thickness Δy) weighs $2 \times 9.8\,\Delta y$ newtons and must be lifted y metres, so it needs work $19.6\,y\,\Delta y$. Total:

$$W = \int_0^{10} 19.6\,y\,dy = 19.6\left[\frac{y^2}{2}\right]_0^{10} = 19.6 \times 50 = 980\ \text{J}.$$

Here the force is constant but the **distance** varies by slice, a different variation from the spring. The slice method does not care which.

19.10.5. Probability densities

For a continuous random quantity, such as the waiting time until the next request hits a server, it makes no sense to ask for the probability of **exactly** 1.000000 seconds. Instead a **probability density function** $p(x)$ describes how probability is spread out, and probabilities are **areas**:

$$P(a \le X \le b) = \int_a^b p(x)\,dx, \qquad p(x) \ge 0, \qquad \int_{-\infty}^{\infty} p(x)\,dx = 1.$$

Example. Waiting times are often modelled by the **exponential density** $p(x) = \lambda e^{-\lambda x}$ for $x \ge 0$. With $\lambda = 0.5$ per second, the probability of waiting at most 1 second is

$$\begin{aligned} P(X \le 1) &= \int_0^1 0.5\,e^{-0.5x}\,dx = \Big[-e^{-0.5x}\Big]_0^1 \\ &= -e^{-0.5} - (-1) = 1 - e^{-0.5} \approx 0.3935. \end{aligned}$$

The integral of the density over $[0, \infty)$ is 1, an improper integral (next section). The whole of Volume 3, Chapter 30 is built on this idea, including the normal (bell) curve, whose integral is the famous non-elementary $\int e^{-x^2}\,dx$.

19.10.6. Accumulated quantities and net change

The car example generalises: **the integral of a rate of change gives the total change**. If $Q'(t)$ is a rate,

$$Q(b) - Q(a) = \int_a^b Q'(t)\,dt.$$

This is FTC Part 2 read physically, and it is sometimes called the **net change theorem**. Water flowing into a tank at $r(t) = 10 + 2t$ litres per minute adds $\int_0^5 (10 + 2t)\,dt = \big[10t + t^2\big]_0^5 = 50 + 25 = 75$ litres in the first 5 minutes. The same pattern turns power (watts) into energy (joules or kWh), a bandwidth curve (bytes per second) into total data transferred, and a births-minus-deaths rate into a population change.

Computer-science connection: cloud bills are integrals

Cloud providers bill some resources by integrating a usage rate over time: GB-hours of storage, vCPU-seconds, "95th percentile" bandwidth. A monitoring system that samples a rate every 60 seconds and reports a total is computing a Riemann sum (usually left or trapezoidal). When a graph in Grafana shows "requests per second" and you ask "how many requests today?", the answer is the area under that graph.



Setting up integrals

- **Units check.** The units of $\int_a^b f(x)\,dx$ are (units of f) × (units of x). Speed in m/s times seconds gives metres; force in N times m gives J. If the units of your integral are wrong, the setup is wrong.
- **Area between curves: find the intersections first.** Using the wrong limits, or not splitting where the curves cross, is the most common error.
- **Volume: square the radius**, $\pi f(x)^2$, not $\pi f(x)$ or $(\pi f(x))^2$.

US curriculum

Area between curves, average value, volumes of revolution (discs and washers, plus "known cross-sections") and motion problems are all in **AP Calculus AB**. Work and probability densities usually appear in college Calculus II, AP Physics C, and AP Statistics (as areas under the normal curve).

19.11. Improper integrals: infinite regions with finite area

19.11.1. The problem

Can a region that stretches out to infinity have finite area? Our definition used a closed interval $[a, b]$, so we need to extend it. And we need to, because probability densities live on unbounded intervals.

19.11.2. The method: take a limit

Integrate up to a finite cut-off b, then let $b \to \infty$:

$$\int_a^{\infty} f(x)\,dx = \lim_{b\to\infty} \int_a^b f(x)\,dx.$$

If the limit exists and is finite, the integral **converges**; otherwise it **diverges**.



Improper integral

An **improper integral** is one where the interval is unbounded (a limit of integration is $\pm\infty$) or the integrand blows up somewhere in the interval. It is defined as a limit of ordinary integrals and is said to **converge** if that limit is a finite number and **diverge** otherwise.

19.11.3. Two worked examples that look alike but are not

Example 1: $\int_1^{\infty} \frac{1}{x^2}\,dx$. *Try it:* ∫1/x² to infinity ; ∫1/x to infinity .

$$\int_1^b x^{-2}\,dx = \left[-\frac{1}{x}\right]_1^b = -\frac{1}{b} - (-1) = 1 - \frac{1}{b}$$

$$\lim_{b\to\infty}\left(1 - \frac{1}{b}\right) = 1.$$

It converges: the infinitely long region has area exactly 1.

Example 2: $\int_1^\infty \frac{1}{x}\,dx$.

$$\int_1^b \frac{1}{x}\,dx = \Big[\ln x\Big]_1^b = \ln b - 0 = \ln b$$
$$\lim_{b\to\infty} \ln b = \infty.$$

It diverges. Both curves fall to zero, and $1/x$ is only a little larger than $1/x^2$ far out, but $1/x$ doesn't fall **fast enough**: the area keeps growing (slowly, like a logarithm) forever.

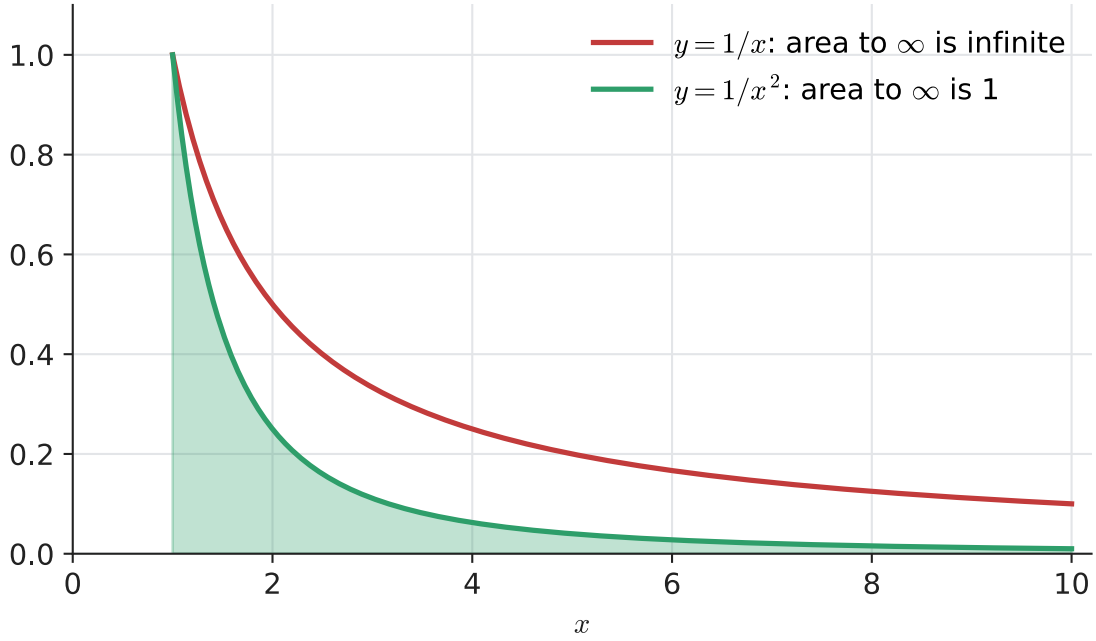


Figure 41. $1/x^2$ encloses finite area out to infinity; $1/x$ does not.

The general rule (the **p-test**): $\int_1^\infty x^{-p}\,dx$ converges exactly when $p > 1$, with value $\frac{1}{p-1}$. The borderline case $p = 1$ diverges.

Example 3: an integrand that blows up. $\int_0^1 \frac{1}{\sqrt{x}}\,dx$. The integrand is infinite at $x = 0$, so cut off at $c > 0$: *Try it:* A spike with finite area ; Evaluate this improper integral .

$$\int_c^1 x^{-1/2}\,dx = \Big[2\sqrt{x}\Big]_c^1 = 2 - 2\sqrt{c} \xrightarrow{c\to 0^+} 2.$$

A spike to infinity with finite area 2. (For $\int_0^1 \frac{1}{x}\,dx$ the same calculation gives $-\ln c \to \infty$: divergent.)



The fake negative area, revisited

An integrand that blows up **inside** the interval is improper even though both limits are finite. $\int_{-1}^1 \frac{1}{x^2}\,dx$ must be split at 0 into two improper integrals, and $\int_0^1 x^{-2}\,dx = \lim_{c\to 0^+}\left(\frac{1}{c} - 1\right) = \infty$, so the whole thing diverges. Always scan the interval for vertical asymptotes before applying the FTC.



Gabriel's horn

Spin $y = 1/x$ for $x \geq 1$ around the x-axis. The volume is $\pi \int_1^\infty x^{-2}\,dx = \pi$, finite. The surface area turns out to be infinite (it is at least $2\pi \int_1^\infty \frac{1}{x}\,dx$). So you could fill the horn with π units of paint but never paint its inside surface. Evangelista Torricelli's 1643 discovery shocked contemporaries who believed infinite objects must have infinite size in every sense.

Try it: Gabriel's horn's volume .

Computer-science connection: the harmonic number

The divergence of $\int_1^\infty \frac{1}{x}\,dx$ has a discrete twin: the harmonic sum $H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$ also grows without bound, like $\ln n$. It shows up in the expected number of comparisons in quicksort (about $2n \ln n$), the coupon-collector problem (nH_n draws to collect all n items), and expected comparisons in randomised algorithms generally. Section 19.14 shows how integrals pin down such sums.

US curriculum

Improper integrals are an **AP Calculus BC** topic (not AB) and part of college Calculus II, where they lead into the convergence tests for infinite series.

19.12. Numerical integration

19.12.1. The problem

Antiderivatives fail us in three common situations:

1. **No elementary antiderivative exists**: $\int_0^1 e^{-x^2}\,dx$, the backbone of the normal distribution.
2. **The function is only known as data**: speed logged by a GPS once a second, power measured every minute.
3. **The function is a black box**: the output of a simulation or another program.

In all three we want a **number**, and the Riemann-sum idea gives it directly. The question is how to get high accuracy from few function evaluations. This is called **numerical integration** or **quadrature**.



Quadrature

Quadrature means numerical approximation of a definite integral. The name comes from the ancient problem of "squaring" a shape: constructing a square with the same area. A **quadrature rule** is a weighted sum $\sum w_i f(x_i)$ of function values at chosen **nodes** x_i.

19.12.2. The trapezoid rule

Intuition. Instead of flat-topped rectangles, join neighbouring points on the curve with straight lines. Each strip becomes a trapezoid, which hugs a smooth curve much better than a rectangle.

Why it works. A trapezoid with parallel sides $f(x_{i-1})$ and $f(x_i)$ and width h has area $\frac{h}{2}\big(f(x_{i-1}) + f(x_i)\big)$. Add the n strips. Every interior point is shared by two neighbouring trapezoids, so it is counted twice; the two endpoints are counted once:

$$T_n = \frac{h}{2}\Big(f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\Big), \qquad h = \frac{b-a}{n}.$$

It is also exactly the average of the left and right Riemann sums.

19.12.3. Simpson's rule

Intuition. Straight lines cannot follow curvature. Parabolas can. Take the points in groups of three, fit the unique parabola through each triple, and integrate the parabola exactly.

Why it works. Shift a triple so the middle node is at 0: nodes $-h, 0, h$ with heights y_0, y_1, y_2. For a parabola $q(x) = Ax^2 + Bx + C$ through them,

$$\int_{-h}^{h} (Ax^2 + Bx + C)\,dx = \frac{2Ah^3}{3} + 2Ch.$$

The Bx term integrates to zero (odd function). From the three points, $q(0) = C = y_1$ and $y_0 + y_2 = q(-h) + q(h) = 2Ah^2 + 2C$, so $2Ah^2 = y_0 + y_2 - 2y_1$. Substitute:

$$\int_{-h}^{h} q(x)\,dx = \frac{h}{3}\big(y_0 + y_2 - 2y_1\big) + 2y_1 h = \frac{h}{3}\big(y_0 + 4y_1 + y_2\big).$$

The middle point gets weight 4, the ends weight 1. Chaining pairs of strips across $[a, b]$ (so n must be **even**) gives weights $1, 4, 2, 4, 2, \ldots, 4, 1$:

$$S_n = \frac{h}{3}\Big(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\Big).$$

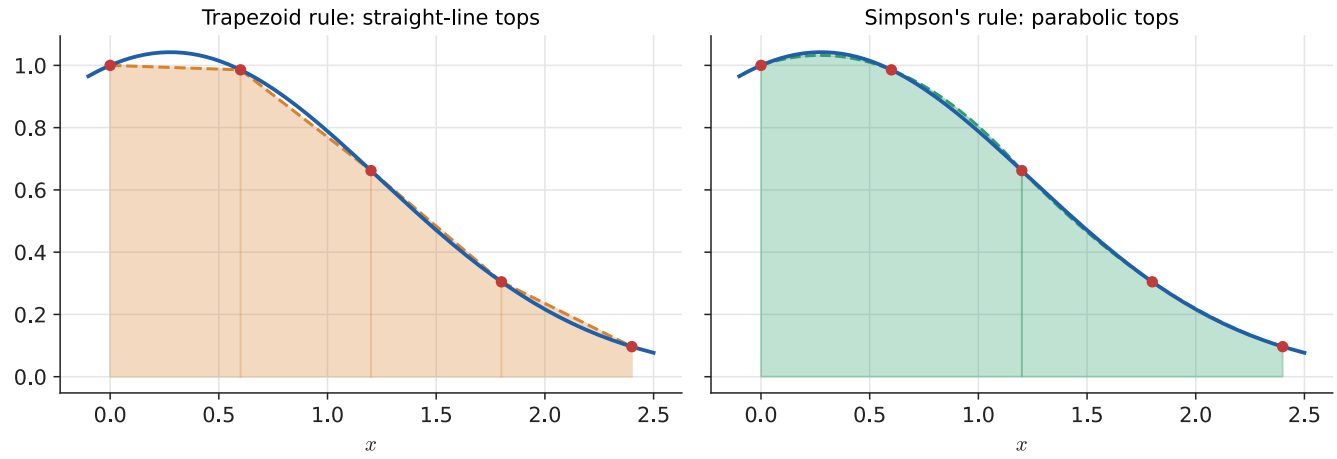


Figure 42. Trapezoids join the sample points with straight lines; Simpson's rule fits a parabola through each consecutive triple.

Example 18. Procedure — trapezoid rule

Input: f, interval $[a, b]$, number of strips n.

Steps:

1. Set $h = (b - a)/n$ and nodes $x_i = a + ih$ for $i = 0, \dots, n$.
2. Evaluate $f(x_0), \dots, f(x_n)$.
3. Form $\frac{1}{2} f(x_0) + f(x_1) + \cdots + f(x_{n-1}) + \frac{1}{2} f(x_n)$.
4. Multiply by h.

Output: $T_n \approx \int_a^b f(x)\,dx$. Error shrinks like h^2: doubling n cuts the error by about 4.

Example 19. Procedure — Simpson's rule

Input: f, interval $[a, b]$, an **even** number of strips n.

Steps:

1. Set $h = (b - a)/n$ and nodes $x_i = a + ih$.
2. Evaluate f at every node.
3. Weight the values $1, 4, 2, 4, \dots, 2, 4, 1$: endpoints 1, odd-indexed nodes 4, interior even-indexed nodes 2.
4. Add the weighted values and multiply by $h/3$.

Output: $S_n \approx \int_a^b f(x)\,dx$. Error shrinks like h^4: doubling n cuts the error by about 16.

19.12.4. Worked example by hand: estimating $\ln 2$

We know $\int_1^2 \frac{1}{x}\,dx = \ln 2 \approx 0.693147$. Pretend we don't, and use $n = 4$, $h = 0.25$. *Try it:* Trapezoid and Simpson for ln 2 .

Nodes and values:

i	0	1	2	3	4
x_i	1	1.25	1.5	1.75	2
$f(x_i) = 1/x_i$	1	0.8	0.666667	0.571429	0.5

Trapezoid:

$$\begin{aligned} T_4 &= 0.25\left(\tfrac{1}{2}(1) + 0.8 + 0.666667 + 0.571429 + \tfrac{1}{2}(0.5)\right) \\ &= 0.25\,(0.5 + 0.8 + 0.666667 + 0.571429 + 0.25) \\ &= 0.25 \times 2.788095 = 0.697024. \end{aligned}$$

Error: $0.697024 - 0.693147 = 0.003877$.

Simpson:

$$\begin{aligned} S_4 &= \frac{0.25}{3}\left(1 + 4(0.8) + 2(0.666667) + 4(0.571429) + 0.5\right) \\ &= \frac{0.25}{3}\,(1 + 3.2 + 1.333333 + 2.285714 + 0.5) \\ &= \frac{0.25}{3} \times 8.319048 = 0.693254. \end{aligned}$$

Error: $0.693254 - 0.693147 = 0.000107$. The **same five function values** give a result 36 times more accurate, just by weighting them more cleverly.

19.12.5. Implementation and error comparison

```python
import math

def trapezoid(f, a, b, n):
    h = (b - a) / n
    total = (f(a) + f(b)) / 2
    for i in range(1, n):
        total += f(a + i * h)
    return h * total

def simpson(f, a, b, n):
    if n % 2:
        raise ValueError("Simpson's rule needs an even number of subintervals")
    h = (b - a) / n
    total = f(a) + f(b)
    for i in range(1, n):
        weight = 4 if i % 2 else 2
        total += weight * f(a + i * h)
    return h * total / 3

exact = 2.0   # integral of sin x from 0 to pi
print(" n    trapezoid error   Simpson error")
for n in [2, 4, 8, 16, 32]:
    et = abs(trapezoid(math.sin, 0, math.pi, n) - exact)
    es = abs(simpson(math.sin, 0, math.pi, n) - exact)
    print(f"{n:3d}   {et:.3e}         {es:.3e}")
```

Try it: Watch the errors shrink .

```
 n    trapezoid error   Simpson error
 2   4.292e-01         9.440e-02
 4   1.039e-01         4.560e-03
 8   2.577e-02         2.692e-04
16   6.430e-03         1.659e-05
32   1.607e-03         1.033e-06
```

Read the columns as ratios. Each doubling of n divides the trapezoid error by about 4 ($0.1039/0.02577 \approx 4.03$) and the Simpson error by about 16 ($2.692 \times 10^{-4}/1.659 \times 10^{-5} \approx 16.2$). That is what "error $\propto h^2$" and "error $\propto h^4$" mean in practice. On a log–log plot those become straight lines of slope -2 and -4; the left Riemann sum has slope -1.

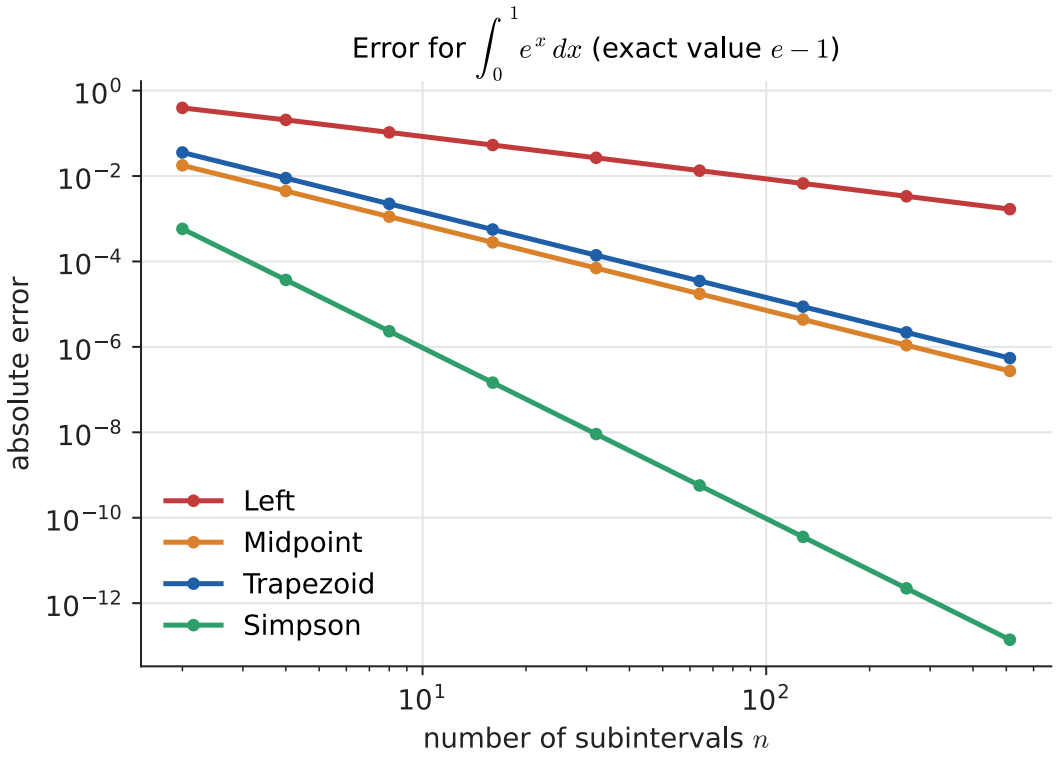


Figure 43. Error versus number of strips (log–log scale) for four rules applied to $\int_0^1 e^x\,dx = e - 1$. Steeper lines converge faster.

19.12.6. Choosing a method

Situation	Choose
Smooth function, can evaluate anywhere, want accuracy cheaply	Simpson (or library adaptive quadrature such as `scipy.integrate.quad`)
Equally spaced data points, odd number of them	Simpson
Data at uneven spacing, or noisy data	Trapezoid (no smoothness to exploit; simple and robust)
Periodic function over a whole period	Trapezoid (it is spectacularly accurate here)
Integrand with a kink or a jump	Split at the bad point, then use either rule on each smooth piece
Many dimensions (say 5 or more)	Monte Carlo (see the sidebar below)
An antiderivative is available	The FTC, exactly

Try it: The log–log error plot .

Where you'd use this: drug exposure. In clinical trials, blood is sampled at a few times after a dose and the **area under the curve** (AUC) of concentration measures the body's total exposure to the drug. With samples at 0, 1, 2, 4 and 8 hours of 0, 8, 10, 6 and 2 mg/L (uneven spacing, so trapezoids), the AUC is $\frac{0+8}{2}(1) + \frac{8+10}{2}(1) + \frac{10+6}{2}(2) + \frac{6+2}{2}(4) = 4 + 9 + 16 + 16 = 45$ mg·h/L. Regulators compare generic and brand-name drugs by exactly this number. The same trapezoid rule turns a GPS speed log into the distance on a running app, and a sampled power trace into the energy a server consumed.



When numerical integration goes wrong

- **Singularities.** Applying Simpson to $\int_0^1 x^{-1/2}\,dx$ tries to evaluate $f(0) = \infty$. Use a substitution to remove the singularity, or a rule designed for it.
- **Oscillation.** $\int_0^{100} \sin(50x)\,dx$ needs many points per wiggle; too few and the samples alias, the same failure as undersampling audio.
- **Infinite intervals.** Map to a finite one (for example $x = 1/t$) or truncate where the tail is provably negligible.
- **Simpson needs even n.** An odd number of strips silently gives garbage in a naive implementation, which is why the code above raises an error.
- **Smaller h is not always better.** Floating-point rounding eventually stops the error falling, as with numerical derivatives in Chapter 17, although integration is far less sensitive than differentiation because it averages rather than subtracts.

Computer-science connection: Monte Carlo integration

Here is a completely different idea. The integral $\int_a^b f(x)\,dx$ equals $(b-a)\times$ (the average value of f). So estimate the average by sampling f at **random** points:

```python
import random, math

def monte_carlo(f, a, b, n, seed=2):
    rng = random.Random(seed)
    total = 0.0
    for _ in range(n):
        total += f(rng.uniform(a, b))   # sample a random point, record height
    return (b - a) * total / n          # width times average height

f = lambda x: math.exp(-x * x)
exact = 0.7468241328124271
for n in [100, 10_000, 1_000_000]:
    est = monte_carlo(f, 0, 1, n)
    print(f"n={n:>9,}  estimate={est:.6f}  error={abs(est - exact):.1e}")
```

```
n=      100  estimate=0.710943  error=3.6e-02
n=   10,000  estimate=0.748450  error=1.6e-03
n=1,000,000  estimate=0.746796  error=2.8e-05
```

The error falls roughly like $1/\sqrt{n}$ (on average; any one run is luckier or unluckier). That is far **slower** than Simpson in one dimension. So why bother? Because the $1/\sqrt{n}$ rate does **not depend on the dimension**. A grid-based rule in 10 dimensions with just 10 points per axis needs 10^{10} evaluations; Monte Carlo doesn't care. Rendering engines integrate light arriving from all directions at every pixel (path tracing), finance integrates over thousands of random market paths, and Bayesian statistics integrates over many parameters. All use Monte Carlo.

A classic toy version estimates π: throw random points into the unit square and count the fraction landing inside the quarter circle, whose area is $\pi/4$.

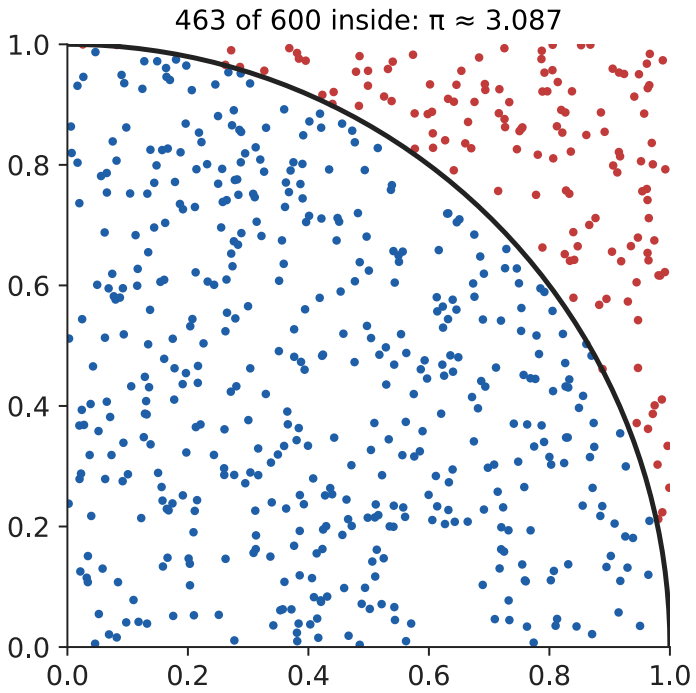


Figure 44. Monte Carlo estimate of π: the fraction of random points inside the quarter circle approximates its area $\pi/4$.

Gaussian quadrature

Trapezoid and Simpson fix the nodes at equal spacing and choose only the weights. Carl Friedrich Gauss asked in 1814: what if you choose the **nodes** as well? With n cleverly placed points (roots of Legendre polynomials) you can integrate every polynomial of degree up to $2n - 1$ exactly. Two well-chosen points beat Simpson's three. Variants of Gaussian quadrature sit inside most scientific integration libraries.

Racing the rules

Choose a function from a menu (smooth, kinked, oscillating, spiking) and a number of strips. Show trapezoids, midpoint rectangles and Simpson parabolas overlaid on the curve, with a live log–log error plot growing as you increase n. Find a function where the trapezoid rule beats Simpson (hint: a periodic function over one full period, or one with a kink).

Try it: A kink slows Simpson · Race the rules

US curriculum

The trapezoid rule (often from a table of data) is tested in **AP Calculus AB**. Simpson's rule appears in many college Calculus II courses and in numerical methods courses; it is no longer on the AP syllabus.

19.13. Differential equations: a first look

19.13.1. The problem

So far the rate of change was a known function of time: $v(t)$ was given, and we integrated to get position. Often, though, the rate depends on the **current amount**. A population grows faster when there are more organisms. A hot cup of coffee cools faster when it is much hotter than the room. A falling object's acceleration depends on its velocity through air resistance. Equations relating a quantity to its own rate of change are called **differential equations**, and they are how science writes down the laws of change.



Differential equation

A **differential equation** is an equation involving an unknown function and its derivatives, such as $\frac{dy}{dt} = ky$. A **solution** is a function that makes the equation true. An **initial condition** such as $y(0) = y_0$ picks out one solution from the family. An **ordinary differential equation** (ODE) has a single independent variable (usually time).

19.13.2. Exponential growth: $dy/dt = ky$

Intuition. The rate of growth is proportional to the current size: double the population, double the births per year. You already know a function with that property: e^t is its own derivative (Chapter 17).

Solving by separation of variables. Move all the y to one side and all the t to the other (treating dy and dt as Leibniz did, which can be justified by the chain rule), then integrate both sides:

$$\begin{aligned} \frac{dy}{dt} &= ky \\ \frac{1}{y}\,dy &= k\,dt && \text{(separate, assuming } y > 0) \\ \int \frac{1}{y}\,dy &= \int k\,dt && \text{(integrate both sides)} \\ \ln y &= kt + C \\ y &= e^{kt+C} = e^{C}e^{kt} && \text{(exponentiate)} \\ y &= y_0\,e^{kt}. && (t=0 \text{ gives } y(0)=e^C\text{; call it } y_0) \end{aligned}$$

Check: $\frac{d}{dt}\left(y_0 e^{kt}\right) = k\,y_0 e^{kt} = ky$. Correct.

Example. Bacteria double every 20 minutes. Then $e^{20k} = 2$, so $k = \frac{\ln 2}{20} \approx 0.03466$ per minute. After 2 hours (6 doubling periods) the population is $e^{120k} = e^{6\ln 2} = 2^6 = 64$ times its initial size. With $k < 0$ the same equation describes radioactive decay, drug elimination and (via Newton's law of cooling, $\frac{dT}{dt} = -k(T - T_{\text{room}})$) that cooling coffee. See Volume 1, Chapter 12 for more on exponential models.

19.13.3. Newton's second law is a differential equation

$F = ma$ says acceleration (the second derivative of position) is force divided by mass:

$$\frac{d^2x}{dt^2} = \frac{F}{m}.$$

When F is constant we integrated twice in Section 19.5 to get the suvat equations. When F depends on position or velocity (springs, drag, gravity between planets), exact solutions are rare and we must step forward numerically.

19.13.4. Euler's method: integrating one small step at a time

Intuition. If you know where you are and how fast you are changing **right now**, assume the rate stays constant for a short time Δt and step forward. Then recompute the rate and repeat. It is a left Riemann sum built on the fly, where each rectangle's height depends on the result so far.

$$y_{n+1} = y_n + \Delta t \cdot \text{rate}(t_n, y_n).$$

Example 20. Procedure — Euler's method

Input: a rate function $\frac{dy}{dt} = g(t, y)$, initial values t_0, y_0, a step size Δt, and an end time T.

Steps:

1. Set $t = t_0, y = y_0$.
2. While $t < T$:
   a. Compute the current rate $r = g(t, y)$.
   b. Update $y \leftarrow y + r\,\Delta t$ (the tangent-line step).
   c. Update $t \leftarrow t + \Delta t$.

Output: the approximate values $y(t)$ at each step. The error at a fixed end time shrinks roughly in proportion to Δt.

Worked example by hand. $\frac{dy}{dt} = 0.5y$, $y(0) = 1$, step $\Delta t = 1$: *Try it:* Step through this example .

$$\begin{aligned} y_1 &= 1 + 1 \times (0.5 \times 1) = 1.5 \\ y_2 &= 1.5 + 1 \times (0.5 \times 1.5) = 2.25 \\ y_3 &= 2.25 + 1 \times (0.5 \times 2.25) = 3.375 \\ y_4 &= 3.375 + 1 \times (0.5 \times 3.375) = 5.0625. \end{aligned}$$

The exact solution is $y(4) = e^2 \approx 7.389$. Each step multiplies by 1.5 instead of $e^{0.5} \approx 1.6487$, so Euler lags behind because it uses the rate at the **start** of each step, while the true rate keeps increasing during the step. Smaller steps fix this, as the figure shows.

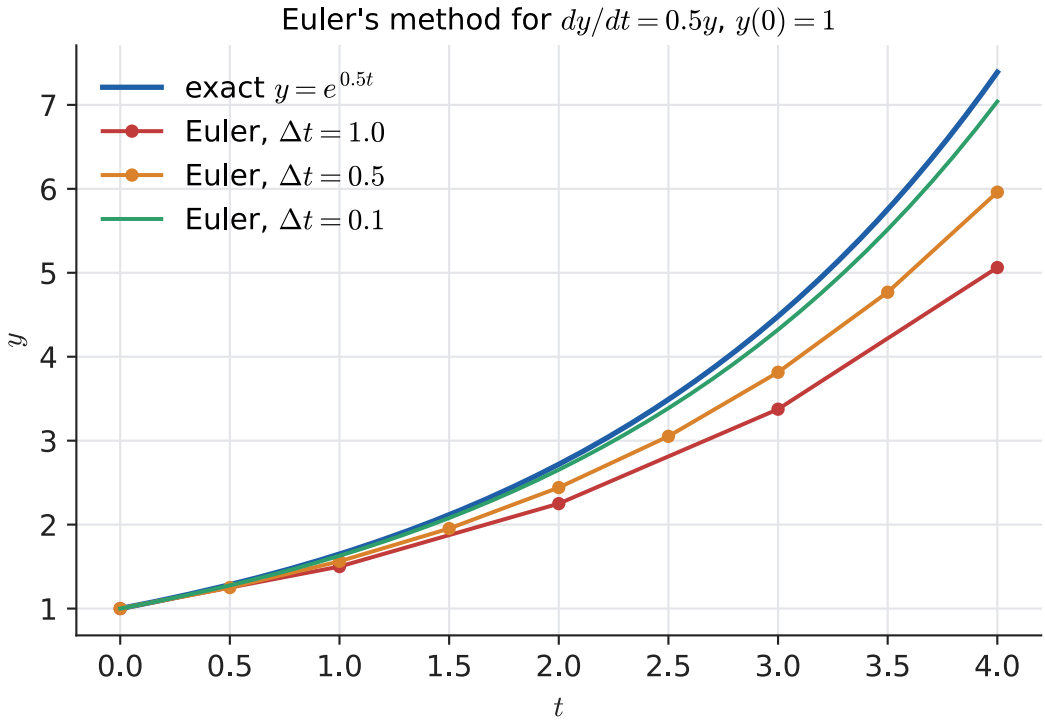


Figure 45. Euler's method for $dy/dt = 0.5y$: big steps lag behind the exact exponential; small steps track it closely.

19.13.5. Mechanics in code: a projectile

For motion we track two quantities, position and velocity, each with its own rate: $\frac{dy}{dt} = v$ and $\frac{dv}{dt} = a = F/m$. *Try it:* Compare the three integrators .

PYTHON

```
g = 9.8                     # m/s^2, downward
dt = 0.1                    # time step (s)
y, v, t = 0.0, 20.0, 0.0    # height (m), upward velocity (m/s), time (s)
while t < 2.0 - 1e-9:
    a = -g                  # Newton: F = m a  =>  a = F/m = -g
    y = y + v * dt          # position changes at rate v
    v = v + a * dt          # velocity changes at rate a
    t = t + dt
print(f"Euler:  t={t:.1f}  y={y:.3f}  v={v:.3f}")
print(f"Exact:  t=2.0  y={20*2 - 0.5*g*4:.3f}  v={20 - g*2:.3f}")
```

```
Euler:  t=2.0  y=21.380  v=0.400
Exact:  t=2.0  y=20.400  v=0.400
```

The velocity is exact (its rate is constant, so a left sum is perfect), but the position is off by nearly a metre: each step uses the velocity from the start of the step, which is always a little too high on the way up.

Computer-science connection: physics engines

Every game physics engine runs a loop like the one above, 60 or more times a second: compute forces, update velocities, update positions. Swap the two update lines (update v first, then use the **new** v to update y) and you get **semi-implicit** (or **symplectic**) **Euler**, which most engines use because it keeps orbits and springs from slowly gaining energy and flying apart. Run that variant and the position comes out at 19.420 m, too **low** by 0.98 m instead of too high. The average of the two runs, 20.400, is exact: averaging the two is the trapezoid rule, and it inspires the **velocity Verlet** integrator used in molecular dynamics. Higher-accuracy ODE solvers such as Runge–Kutta methods are developed in Chapter 32.

Try it: Euler vs symplectic vs Verlet .



A rate that depends on itself

Draw a **slope field** for $dy/dt = g(t, y)$: a tiny line segment at each grid point showing the slope there. Click a starting point and watch Euler's method trace a path through the field with an adjustable Δt, alongside the exact solution where known. Try $dy/dt = 0.5y$, $dy/dt = -y$, and the logistic equation $dy/dt = y(1 - y)$.

Try it: Open the slope-field lab



Euler on large steps

Euler's method can be not just inaccurate but **unstable**. For $dy/dt = -3y$ with $\Delta t = 1$, each step multiplies y by $1 - 3 = -2$: the "decaying" solution flips sign and dodoubles every step, exploding instead of decaying. A physics engine with too large a time step on a stiff spring does exactly this, which is why objects sometimes jitter and fly off into space in games.

Try it: Watch Euler explode .

19.13.6. Where you'd use this

- **Everyday: caffeine.** Caffeine is eliminated roughly exponentially, $\frac{dC}{dt} = -kC$, with a half-life of about 5 hours. A 100 mg coffee at 4 pm leaves $100 \times 2^{-7/5} \approx 38$ mg in your system at 11 pm: more than a third of a cup at bedtime.
- **Epidemiology.** The SIR model tracks susceptible, infected and recovered people with three coupled differential equations (for example $\frac{dI}{dt} = \beta SI - \gamma I$). Public-health forecasts during the COVID-19 pandemic were refined versions of this, solved by step-by-step methods like Euler's.
- **Computing.** Every physics engine, weather model, circuit simulator (SPICE) and orbital-mechanics program is a differential-equation solver at heart: it knows the rates (forces, currents, winds) and integrates them forward in small time steps, as in the projectile code above.

US curriculum

Separable differential equations, exponential growth and decay, and slope fields are in **AP Calculus AB**; Euler's method and the logistic equation are in **AP Calculus BC**. A full differential equations course is usually a second-year college course.

19.14. Sums and integrals in algorithm analysis

19.14.1. The problem

How many times does the inner statement run in this loop? *Try it:* The loop as a staircase .

```python
count = 0
n = 8
for i in range(1, n + 1):
    for j in range(i):        # inner loop runs i times
        count += 1
print(count)                  # 36
```

The exact answer is $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$, which is 36 for $n = 8$. For algorithm analysis we usually care about growth: $\frac{n(n+1)}{2} \approx \frac{n^2}{2}$, so the loop is $O(n^2)$. Notice that $\int_0^n x\,dx = \frac{n^2}{2}$ gives the leading term immediately.

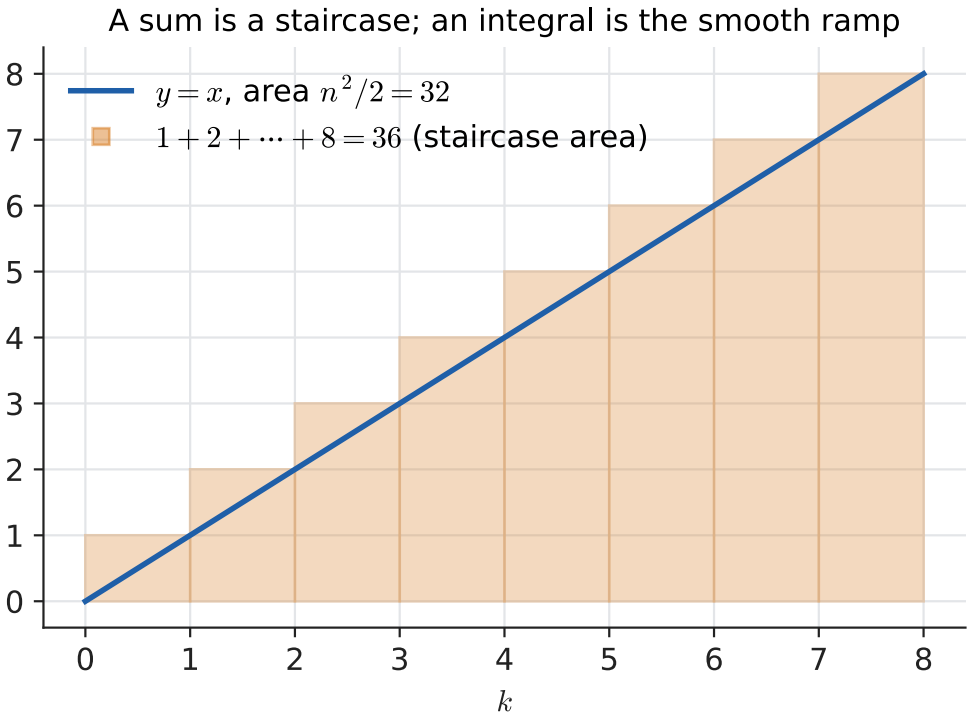


Figure 46. A sum is a staircase of unit-width rectangles; the integral is the smooth ramp beneath. They differ by a lower-order amount.

19.14.2. Why it works

A sum $\sum_{k=1}^{n} f(k)$ **is** a Riemann sum with width 1. If f is increasing, comparing each unit rectangle with the area under the curve on either side gives the **integral bounds**

$$\int_0^n f(x)\,dx \;\le\; \sum_{k=1}^{n} f(k) \;\le\; \int_1^{n+1} f(x)\,dx.$$

(For decreasing f the inequalities flip.) The sum is trapped between two integrals that are easy to compute, so it grows at the same rate as the integral.

19.14.3. Worked examples

Sum	Integral estimate	Where it appears
$\sum_{k=1}^{n} k$	$\int_0^n x\,dx = \frac{n^2}{2}$	Nested loops, insertion sort worst case
$\sum_{k=1}^{n} k^2$	$\int_0^n x^2\,dx = \frac{n^3}{3}$	Triple loops over triangles, naive Gaussian elimination
$\sum_{k=1}^{n} \frac{1}{k}$	$\int_1^n \frac{dx}{x} = \ln n$	Quicksort, coupon collector, harmonic series
$\sum_{k=1}^{n} \ln k = \ln(n!)$	$\int_1^n \ln x\,dx = n\ln n - n + 1$	Lower bound $\Omega(n \log n)$ for comparison sorting

For the harmonic sum, $1/x$ is decreasing, so the bounds read $\ln(n+1) \le H_n \le 1 + \ln n$. For $n = 1000$ that is $6.909 \le H_{1000} \le 7.908$; the true value is about 7.485. *Try it:* Bound the harmonic sum .

The last row is worth a moment. $\int_1^n \ln x \, dx$ uses the integration-by-parts result from Section 19.8, and gives $\log_2(n!) \approx n \log_2 n$. Since a comparison sort must distinguish all $n!$ orderings and each comparison gives one bit, no comparison sort can beat roughly $n \log_2 n$ comparisons. An integration-by-parts exercise proves a fundamental limit of computer science. *Try it:* ln n! against an integral .

Computer-science connection: discrete and continuous calculus

Sums are to integrals what differences are to derivatives. The **finite difference** $\Delta f(k) = f(k+1) - f(k)$ plays the role of f', and "summation by parts" mirrors integration by parts. Knuth's *Concrete Mathematics* (the name blends CONtinuous and disCRETE) develops this "finite calculus" for exactly the sums that algorithm analysis produces. When a recurrence or a loop count gets messy, replacing the sum with an integral is often the fastest way to the right big-O.

19.15. Exercises: applications, improper integrals and numerics

1. **Recognition.** Which of these improper integrals converge? $\int_1^\infty x^{-3}\,dx$, $\int_1^\infty x^{-1/2}\,dx$, $\int_0^1 x^{-1}\,dx$.
2. **Application.** Find the area between $y = 2x$ and $y = x^2$. *Try it:* Area between 2x and x² .
3. **Application.** Find the average value of x^2 on $[0, 3]$.
4. **Application.** Find the volume generated by rotating $y = x$, $0 \le x \le 3$, about the x-axis, and check it against the cone formula $\frac{1}{3}\pi r^2 h$. *Try it:* Spin y = x: a cone .
5. **Application.** Approximate $\int_0^2 x^3\,dx$ with the trapezoid rule and with Simpson's rule, both with $n = 2$. Explain what you notice. *Try it:* Simpson is exact for cubics .
6. **Problem solving.** A population of 500 grows according to $dP/dt = 0.02P$ (time in years). What is it after 50 years? Use Euler's method with $\Delta t = 25$ years and compare.
7. **Problem solving.** Use integral bounds to show that $\sum_{k=1}^{n} \sqrt{k}$ grows like $\frac{2}{3}n^{3/2}$. *Try it:* Σ√k against ⅔n^(3/2) .

▼ **Answers**

1. $\int_1^\infty x^{-3}\,dx = \lim_{b\to\infty}\left[-\frac{1}{2x^2}\right]_1^b = \frac{1}{2}$: converges ($p = 3 > 1$).
   $\int_1^\infty x^{-1/2}\,dx = \lim\left[2\sqrt{x}\right]_1^b = \infty$: diverges ($p = \frac{1}{2} \le 1$). $\int_0^1 x^{-1}\,dx = \lim_{c\to 0^+}(-\ln c) = \infty$: diverges.
2. Intersections: $2x = x^2 \Rightarrow x = 0, 2$. On $(0, 2)$, $2x \ge x^2$. Area $= \int_0^2 (2x - x^2)\,dx = \left[x^2 - x^3/3\right]_0^2 = 4 - 8/3 = 4/3$.
3. $\frac{1}{3}\int_0^3 x^2\,dx = \frac{1}{3}\cdot\frac{27}{3} = 3$.

4. $V = \pi \int_0^3 x^2\,dx = \pi \cdot 9 = 9\pi$. The solid is a cone with $r = 3$, $h = 3$: $\frac{1}{3}\pi \cdot 9 \cdot 3 = 9\pi$. They agree.

5. $h = 1$, values $0, 1, 8$. Trapezoid: $1 \cdot (0/2 + 1 + 8/2) = 5$. Simpson: $\frac{1}{3}(0 + 4 \cdot 1 + 8) = 4$. The exact value is $\left[x^4/4\right]_0^2 = 4$: Simpson is **exact**. Although built from parabolas, Simpson's rule happens to integrate cubics exactly too (the cubic error term cancels by symmetry), which is why its error is $O(h^4)$ rather than $O(h^3)$.

6. Exact: $P = 500e^{0.02 \times 50} = 500e \approx 1359.1$. Euler: $P_1 = 500 + 25(0.02)(500) = 750$, $P_2 = 750 + 25(0.02)(750) = 1125$. Euler underestimates (each step multiplies by 1.5 instead of $e^{0.5} \approx 1.649$).

7. $\sqrt{x}$ is increasing, so $\int_0^n \sqrt{x}\,dx \le \sum_{k=1}^{n} \sqrt{k} \le \int_1^{n+1} \sqrt{x}\,dx$, that is $\frac{2}{3}n^{3/2} \le \sum \le \frac{2}{3}\left((n+1)^{3/2} - 1\right)$. Both bounds are $\frac{2}{3}n^{3/2}$ to leading order.

19.16. Interactive tools

Browser tools on the companion website that go with this chapter. Each example link in the chapter opens one of them with that example already set up.

- Calculus, visually
- Graphing calculator
- Symbolic algebra
- ODE solver and phase portraits

19.17. Mastery check

1. Evaluate $\int_1^4 (\sqrt{x} + 2)\,dx$.
2. Estimate $\int_0^4 x^2\,dx$ with a midpoint sum using $n = 2$, and compare with the exact value.
3. Find $\int x \sin(x^2)\,dx$.
4. Evaluate $\int_0^1 xe^{-x}\,dx$.
5. Find $\frac{d}{dx}\int_2^x \sqrt{1 + t^3}\,dt$.
6. Evaluate $\int_0^\infty e^{-2x}\,dx$, and decide whether $\int_1^\infty \frac{1}{\sqrt{x}}\,dx$ converges.
7. A particle moves with velocity $v(t) = 3t^2 - 12$ m/s for $0 \le t \le 3$. Find its displacement and the total distance travelled.
8. You have a smooth function's values at 11 equally spaced points and need its integral accurately. Which rule do you use, and why?

Decision rule: All correct without hesitation → move on to Chapter 20 (or branch to Volume 3, Chapter 30). Stuck on 1–2 → reread Section 19.4 and Section 19.5. Stuck on 3–4 → rework Section 19.7 and Section 19.8. Stuck on 6–8 → reread Section 19.11, Section 19.10 and Section 19.12. Stuck on more than three → work the section exercises first, and check the derivative rules in Chapter 17, since every integral here is a derivative read backwards.

Try it: Midpoint sum for ∫₀⁴ x² ; ∫e^(−2x) to infinity .

▼ **Mastery check answers**

1. $\int_1^4 (x^{1/2} + 2)\,dx = \left[\frac{2}{3}x^{3/2} + 2x\right]_1^4 = \left(\frac{16}{3} + 8\right) - \left(\frac{2}{3} + 2\right) = \frac{14}{3} + 6 = \frac{32}{3} \approx 10.667$.
2. $\Delta x = 2$, midpoints 1 and 3: $2(1^2 + 3^2) = 20$. Exact: $\left[x^3/3\right]_0^4 = 64/3 \approx 21.333$. The midpoint sum underestimates by about 1.33 (the curve is concave up, so each midpoint rectangle misses a little more than it gains).
3. $u = x^2$, $x\,dx = \frac{1}{2}du$: $\frac{1}{2}\int \sin u\,du = -\frac{1}{2}\cos(x^2) + C$.
4. Parts with $u = x$, $dv = e^{-x}dx$, $v = -e^{-x}$:
   $\left[-xe^{-x}\right]_0^1 + \int_0^1 e^{-x}\,dx = -e^{-1} + \left[-e^{-x}\right]_0^1 = -e^{-1} - e^{-1} + 1 = 1 - \frac{2}{e} \approx 0.2642$.
5. By FTC Part 1: $\sqrt{1 + x^3}$. The lower limit 2 does not matter.
6. $\lim_{b\to\infty}\left[-\frac{1}{2}e^{-2x}\right]_0^b = 0 + \frac{1}{2} = \frac{1}{2}$. $\int_1^\infty x^{-1/2}\,dx$ diverges ($p = \frac{1}{2} \le 1$; $2\sqrt{b} \to \infty$).
7. Displacement: $\left[t^3 - 12t\right]_0^3 = 27 - 36 = -9$ m (ends 9 m behind the start). Distance: $v = 0$ at $t = 2$. On $[0, 2]$, $\int v\,dt = 8 - 24 = -16$, so 16 m backwards. On $[2, 3]$, $\int v\,dt = (27 - 36) - (8 - 24) = 7$ m forwards. Total distance $16 + 7 = 23$ m.

8. Simpson's rule with $n = 10$ strips (11 points gives an even number of strips). The function is smooth, so Simpson's $O(h^4)$ error beats the trapezoid rule's $O(h^2)$ at no extra cost: same data, different weights.

US curriculum

This chapter as a whole covers the integral half of **AP Calculus AB** and most of the additional integration content of **AP Calculus BC** (integration by parts, improper integrals, Euler's method). A US student usually meets it in grade 11 or 12, or in the first year of college. UK readers will recognise the A-level Pure "integration" material plus the Mechanics use of integration for variable acceleration.

Chapter 20. Multivariable Calculus and the Gradient

Everything in the calculus chapters so far has had one input. A car's position depends on time; a curve's height depends on x. But the quantities you actually care about rarely depend on just one thing. The temperature in a room depends on where you stand in three dimensions. A neural network's error depends on millions of weights at once. This chapter extends derivatives to functions of several variables. The central character is the gradient, ∇f, a vector that tells you which way is uphill and how steep it is. By the end, you should be able to look at $\nabla f(\mathbf{x})$ in a machine-learning paper and read it as fluently as a line of code.

Why this chapter matters: real decisions trade off several things at once. A recipe's cost depends on every ingredient's price; a building's heat loss on wall area, insulation and temperature difference; a model's error on every one of its weights. Multivariable calculus tells you which input matters most right now (partial derivatives), which way to adjust them all together (the gradient), and how to find the best setting when you have a budget to respect (Lagrange multipliers).

Where this chapter sits

Prerequisites: derivatives and the chain rule (Chapter 17), optimisation with one variable (Chapter 18), definite integrals (Chapter 19), functions and graphs (Volume 1, Chapter 11), coordinates and distance (Volume 1, Chapter 10).

Helpful but not required: the dot product, which this chapter introduces briefly and Chapter 26 develops in full.

Unlocks: gradient-based optimisation and machine learning (Volume 3, Chapter 33), numerical optimisation and simulation (Chapter 32), joint probability densities (Volume 3, Chapter 30), and the Hessian's role in eigenvalue analysis (Chapter 28).

20.1. Functions of several variables

20.1.1. The problem

A hiker stands on a hillside holding a map. The map shows position as two numbers: east–west (x) and north–south (y). At every position the ground has a height. So height is a function of *two* inputs, $h(x, y)$.

The hiker's questions are calculus questions. How steep is it here? If I walk north-east, will I climb or descend, and how quickly? Which direction climbs fastest? Where is the summit? With one input there was only one direction to move in, forwards or backwards along the x-axis. With two inputs there are infinitely many directions. The single-variable derivative has to be extended to handle that.

20.1.2. Intuition

A function of two variables takes a point (x, y) in the plane and returns a number. You can picture that in three ways, and you should be able to move between them:

- **A surface.** Draw the plane flat and put a point at height $z = f(x, y)$ above each (x, y). The points together form a landscape.
- **A contour plot.** Look straight down on the landscape and draw curves joining points of equal height, exactly like the contour lines on a hiking map.
- **A heat map.** Colour each point of the plane by its height. Weather maps of temperature work this way.

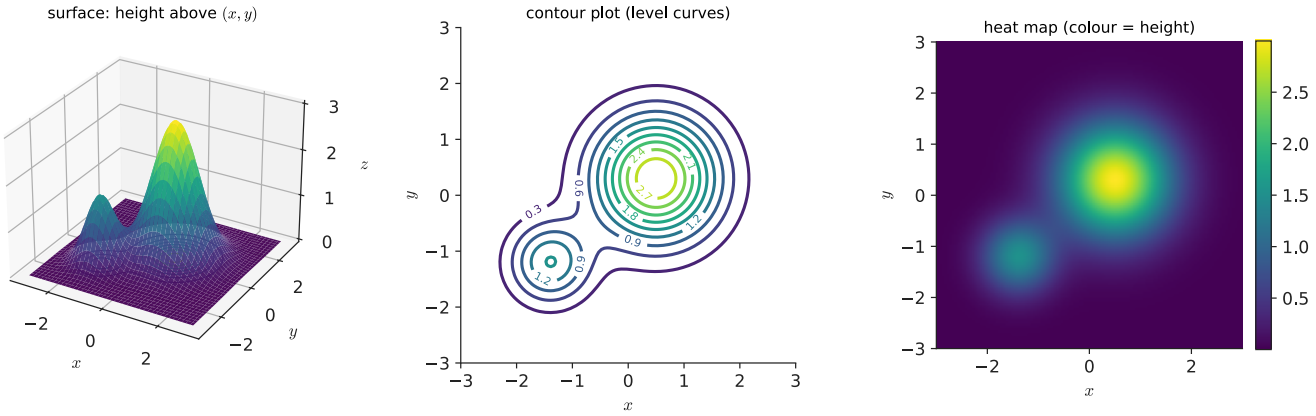


Figure 47. One function, three pictures. Left: the surface $z = f(x, y)$. Centre: contour lines (curves of constant height). Right: a heat map. The function is a large hill with a smaller hill to the south-west.

Contour plots are often more useful than surfaces. A surface hides its back side and is hard to read numerically. A contour plot shows everything at once, and the spacing of the contours tells you about steepness: **closely packed contours mean steep ground, widely spaced contours mean gentle ground**. Hold on to that sentence. It comes back when we meet the gradient.

20.1.3. Vocabulary and notation



Function of several variables

A **function of several variables** (or **multivariable function**) takes more than one input number and returns one output number. For example, $f(x, y) = x^2 + 3y^2$ takes two inputs, and $T(x, y, z)$ could give the temperature at a point in a room.

A **scalar-valued** function returns a single number. (A function returning several numbers, such as a position in space, is **vector-valued**. We'll meet one when we discuss the Jacobian.)



$\mathbb{R}^2$, $\mathbb{R}^n$, and bold vectors

$\mathbb{R}$ is the set of real numbers. $\mathbb{R}^2$ is the set of all ordered pairs (x, y) of real numbers, i.e. the plane. $\mathbb{R}^n$ is the set of all ordered lists of n real numbers, $(x_1, x_2, \ldots, x_n)$.

So $f : \mathbb{R}^2 \to \mathbb{R}$ reads "f takes a pair of reals to a real".

When there are many inputs we bundle them into one object written in **bold**: $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, and write $f(\mathbf{x})$ instead of $f(x_1, \ldots, x_n)$. In handwriting, bold becomes an underline or an arrow: $\vec{x}$. A model with a million weights is just a function $L : \mathbb{R}^{1\,000\,000} \to \mathbb{R}$.



Surface, level curve, contour

The **graph** (or **surface**) of $f(x, y)$ is the set of points (x, y, z) with $z = f(x, y)$.

A **level curve** of f at level c is the set of points in the plane where $f(x, y) = c$. A **contour plot** draws several level curves at evenly spaced levels. With three inputs, the same idea gives **level surfaces** $f(x, y, z) = c$.

20.1.4. Worked example: level curves of a bowl

Take $f(x, y) = x^2 + 3y^2$. What do its level curves look like? *Try it:* See the bowl's elliptical level curves .

Set $f(x, y) = c$ for a few values of c:

$$
\begin{aligned}
c = 0: &\quad x^2 + 3y^2 = 0 \;\Rightarrow\; (x, y) = (0, 0) && \text{(a single point: the bottom)} \\
c = 3: &\quad x^2 + 3y^2 = 3 \;\Rightarrow\; \frac{x^2}{3} + \frac{y^2}{1} = 1 && \text{(divide by 3)} \\
c = 12: &\quad x^2 + 3y^2 = 12 \;\Rightarrow\; \frac{x^2}{12} + \frac{y^2}{4} = 1 && \text{(divide by 12)}
\end{aligned}
$$

Each level curve with $c > 0$ is an ellipse. Along the x-axis, the $c = 3$ ellipse reaches $x = \pm\sqrt{3} \approx \pm 1.73$. Along the y-axis it reaches only $y = \pm 1$. The ellipses are wider than they are tall, so the bowl rises more steeply in the y direction than in the x direction. That matches the formula: the coefficient 3 on y^2 makes y count for more. Negative values of c give no level curve at all, because the bowl never goes below zero.

We'll use this bowl later as the test case for gradient descent.



Contour spacing is not height

A contour plot shows height only through the labels on the curves. Two plots can look identical while describing hills of very different heights if their levels are spaced differently. Always check the contour interval. And remember that each curve is a set of points at one height. Moving *along* a contour does not change f at all.

Applications of multivariable functions

- **Everyday:** body-mass index is a function of two variables, $B(m, h) = m/h^2$ (mass in kg, height in m): 70 kg and 1.75 m give $B = 70/1.75^2 \approx 22.9$. Weather maps (temperature and pressure over latitude and longitude), topographic maps, and the cost of a trip as a function of distance and fuel price are all functions of two or more inputs.
- **Science:** the ideal gas law $P = nRT/V$ makes pressure a function of temperature and volume. Potential energy over a region of space.
- **Computing:** an image is a function $(\text{row}, \text{col}) \mapsto \text{brightness}$. The loss of a machine-learning model is a function of all of its parameters.



Explore a surface

Type a formula $f(x, y)$ and see three linked views: the rotatable surface, the contour plot and the heat map. Hover over a point in any view to highlight it in the other two. Change the contour spacing and watch steep regions show up as bunched lines.

Try it: Open the surface explorer

US curriculum

Functions of two variables, surfaces and contour plots are not part of the standard US high-school sequence. They appear in **Calculus III** (Multivariable Calculus), usually a college sophomore course. Some high schools offer it as a post-AP course after **AP Calculus BC**. Students do see contour maps informally in Earth Science and Geography.

20.2. Partial derivatives

20.2.1. The problem

On the hill $h(x, y)$, "the slope" is ambiguous. The ground can climb steeply to the north and be level to the east at the same point. The derivative of Chapter 17 asked how the output changes when the single input changes. Now there are two inputs, so we have to ask a more specific question: how does the output change when *one* input changes and the others stay put?

20.2.2. Intuition: freeze everything else

Here is the key move, and it's simpler than it sounds. **Freeze all inputs but one.** Then the function really is a function of one variable, and you already know how to differentiate it.

Geometrically, freezing $y = 2$ slices the surface with the vertical plane $y = 2$. The slice is an ordinary curve, and its slope at a point is the **partial derivative with respect to x**. Freezing x instead slices in the other direction and gives the partial derivative with respect to y.

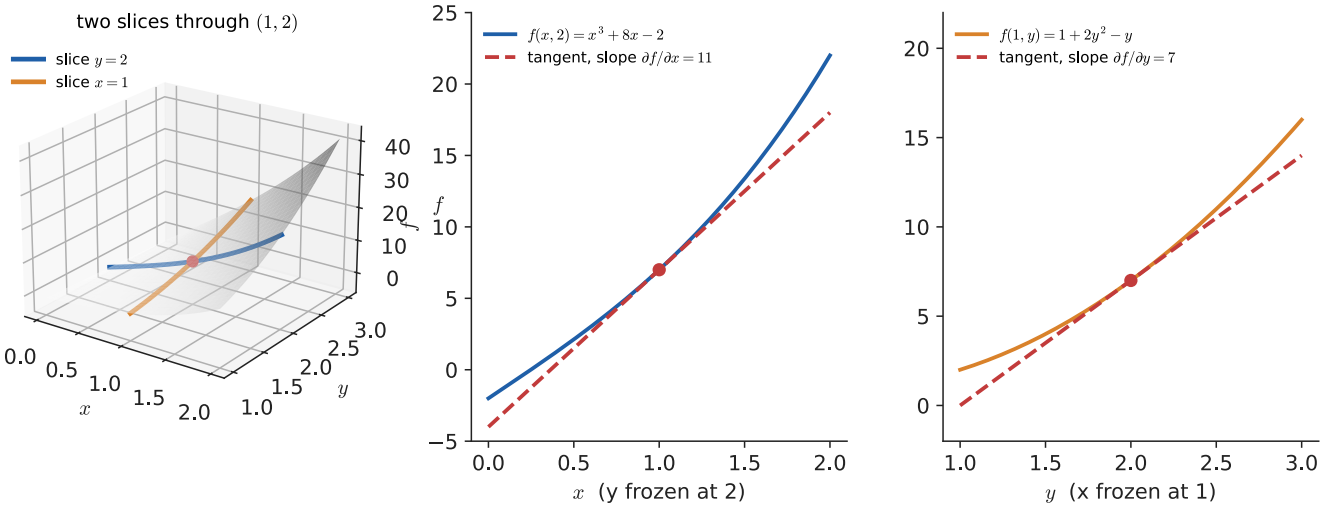


Figure 48. Partial derivatives as slopes of slices. Left: the surface $f(x, y) = x^3 + 2xy^2 - y$ cut by the planes $y = 2$ (blue) and $x = 1$ (orange), which meet at $(1, 2)$. Centre and right: each slice is an ordinary curve with an ordinary tangent line.

20.2.3. Why it works: the definition

The partial derivative is the single-variable derivative applied to a slice, so its definition is the familiar limit with only one input nudged:

$$\frac{\partial f}{\partial x}(a, b) = \lim_{h \to 0} \frac{f(a + h,\, b) - f(a,\, b)}{h}, \qquad \frac{\partial f}{\partial y}(a, b) = \lim_{h \to 0} \frac{f(a,\, b + h) - f(a,\, b)}{h}.$$

Look at what changes in each numerator. In the first, only the first argument moves ($a \to a + h$) and b is untouched. In the second, only b moves. Nothing else is new. Every rule from Chapter 17 (power, product, quotient, chain) still works. The one new habit is to **treat the frozen variables as constants**.

20.2.4. Vocabulary and notation

Partial derivative and the symbol ∂

The **partial derivative** of f with respect to x is the rate of change of f as x varies with every other input held fixed.

Notations, all meaning the same thing:

$$\frac{\partial f}{\partial x}, \qquad f_x, \qquad \partial_x f, \qquad D_x f.$$

The curly ∂ is read "partial" (or "del", or "dee"). It replaces the straight d of $\frac{df}{dx}$ as a reminder that other variables exist and are being held fixed. To evaluate at a point, write $\frac{\partial f}{\partial x}(1, 2)$ or $f_x(1, 2)$.

Second and mixed partial derivatives

Differentiating a partial derivative again gives a **second partial derivative**:

$$f_{xx} = \frac{\partial^2 f}{\partial x^2}, \qquad f_{yy} = \frac{\partial^2 f}{\partial y^2}, \qquad f_{xy} = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) = \frac{\partial^2 f}{\partial y\,\partial x}.$$

f_{xy} and f_{yx} are **mixed partials**. For every function you will meet in practice (any function whose second partials are continuous), **Clairaut's theorem** says they are equal: $f_{xy} = f_{yx}$. The order of differentiation does not matter.

20.2.5. The method

Example 21. Procedure — computing a partial derivative

Input: a formula $f(x_1, \ldots, x_n)$ and the variable x_i to differentiate with respect to.

Steps:

1. Mentally replace every variable except x_i by a constant. It can help to circle them or rename them $a, b, \ldots$.
2. Differentiate with respect to x_i using the ordinary single-variable rules. A term with no x_i in it is a constant, so its derivative is 0. A constant factor multiplying x_i stays as a factor.
3. Restore the frozen variables. Substitute a point if one is given.

Output: the partial derivative $\partial f / \partial x_i$, as a formula or as a number at a point.

20.2.6. Worked example

Find both partial derivatives of *Try it:* Partials as slices at (1, 2) .

$$f(x, y) = x^3 + 2xy^2 - y$$

at the point $(1, 2)$.

With respect to x (freeze y, so y^2 behaves like a number such as 4):

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{\partial}{\partial x}(x^3) + \frac{\partial}{\partial x}(2y^2 \cdot x) - \frac{\partial}{\partial x}(y) && \text{(differentiate term by term)} \\ &= 3x^2 + 2y^2 \cdot 1 - 0 && \text{($2y^2$ is a constant factor; y is a constant)} \\ &= 3x^2 + 2y^2. \end{aligned}$$

With respect to y (freeze x):

$$\begin{aligned} \frac{\partial f}{\partial y} &= \frac{\partial}{\partial y}(x^3) + \frac{\partial}{\partial y}(2x \cdot y^2) - \frac{\partial}{\partial y}(y) && \text{(term by term)} \\ &= 0 + 2x \cdot 2y - 1 && \text{(x^3 is constant; power rule on y^2)} \\ &= 4xy - 1. \end{aligned}$$

At $(1, 2)$:

$$f_x(1,2) = 3(1)^2 + 2(2)^2 = 3 + 8 = 11,$$
$$f_y(1,2) = 4(1)(2) - 1 = 8 - 1 = 7.$$

Interpretation. Standing at $(1, 2)$, where $f(1, 2) = 1 + 8 - 2 = 7$, a small step Δx east changes f by about $11\,\Delta x$. A small step Δy north changes it by about $7\,\Delta y$. These are the slopes of the two dashed tangent lines in the slice figure above.

Second partials (we'll need them for the Hessian):

$$f_{xx} = 6x, \qquad f_{yy} = 4x, \qquad f_{xy} = \frac{\partial}{\partial y}\left(3x^2 + 2y^2\right) = 4y, \qquad f_{yx} = \frac{\partial}{\partial x}\left(4xy - 1\right) = 4y.$$

The mixed partials agree, as Clairaut promised. At $(1, 2)$: $f_{xx} = 6$, $f_{yy} = 4$, $f_{xy} = 8$.

20.2.7. Alternative approach: numerical partial derivatives

When you have code for f but no formula, or you just want to check your algebra, approximate each partial with a **central difference**. Nudge one input up and down by a small h and leave the others alone:

$$\frac{\partial f}{\partial x}(a, b) \approx \frac{f(a + h, b) - f(a - h, b)}{2h}.$$

PYTHON

```python
def f(x, y):
    return x**3 + 2*x*y**2 - y

def partial_x(f, x, y, h=1e-6):
    return (f(x + h, y) - f(x - h, y)) / (2*h)   # y held fixed

def partial_y(f, x, y, h=1e-6):
    return (f(x, y + h) - f(x, y - h)) / (2*h)   # x held fixed

print(round(partial_x(f, 1, 2), 6), round(partial_y(f, 1, 2), 6))
```

```
11.0 7.0
```

Choosing a method. Use the symbolic rules whenever you have a formula. They give exact answers and formulas you can reuse. Use finite differences to *check* symbolic or hand-coded derivatives, or when f is a black box. Avoid them as the main tool when there are many inputs: each partial costs two function evaluations, so a function of a million variables would need two million evaluations per gradient. Chapter 32 explains why h can be neither too large nor too small.



What counts as a constant

The most common slip is differentiating a frozen variable anyway. With respect to x:

- $\frac{\partial}{\partial x}(y^2) = 0$, not $2y$. y^2 is a constant.
- $\frac{\partial}{\partial x}(xy^2) = y^2$, not $2xy$. y^2 is a constant *factor*.
- $\frac{\partial}{\partial x}(e^{xy}) = y\,e^{xy}$. Use the chain rule, with y as the constant multiplier inside.

A useful self-check: in $\partial f/\partial x$, any term without an x must disappear.

Applications of partial derivatives

- **Health:** for BMI $B = m/h^2$, $\partial B/\partial m = 1/h^2 \approx 0.33$ per kg and $\partial B/\partial h = -2m/h^3 \approx -26$ per metre at 70 kg and 1.75 m. So gaining 1 kg raises BMI by about 0.33, while a height measurement that is 1 cm too short raises it by about 0.26. A careless tape measure matters almost as much as a kilogram.
- **Economics:** the **marginal product of labour** is $\partial Q/\partial L$ for a production function $Q(L, K)$, meaning the extra output from one more worker with capital fixed. For the classic Cobb–Douglas form $Q = 10L^{1/2}K^{1/2}$ with 100 workers and 400 units of capital, $Q = 2000$ and $\partial Q/\partial L = 5\sqrt{K/L} = 10$: one more worker adds about 10 units of output, while $\partial Q/\partial K = 5\sqrt{L/K} = 2.5$ per unit of capital.
- **Physics:** in thermodynamics, $(\partial P/\partial T)_V$ is how pressure rises with temperature at constant volume. The subscript V states explicitly what is held fixed.
- **Sensitivity analysis:** in any model, $\partial(\text{output})/\partial(\text{parameter})$ measures how much one assumption matters while the others stay put.



Where did ∂ come from?

The curly ∂ is a rounded letter d. Adrien-Marie Legendre used it in 1786 for partial derivatives, then abandoned it. Carl Jacobi revived it in 1841, and it has stuck ever since. Typesetters and programmers call it "del", "partial" or "curly d". In LaTeX it is `\partial`.

US curriculum

Partial derivatives are a **Calculus III** (Multivariable Calculus) topic, beyond AP Calculus BC. A student who has finished AP Calculus AB already has every rule needed. The only new idea is "hold the others constant".

20.2.8. Exercises

1. Find f_x and f_y for $f(x, y) = 5x^2y^3$.
2. Find g_x and g_y for $g(x, y) = e^{xy} + \ln y$.
3. Find all three first partial derivatives of $h(x, y, z) = xyz + z^2$.

4. Verify that $f_{xy} = f_{yx}$ for $f(x, y) = x^2 y^3$.
5. A rectangular box has volume $V(l, w, h) = lwh$. Compute $\partial V/\partial h$ at $(l, w, h) = (3, 2, 5)$ and explain what it means physically.

▼ **Answers**

1. $f_x = 10xy^3$ ($5y^3$ is a constant factor; differentiate x^2). $f_y = 15x^2y^2$ ($5x^2$ is constant; differentiate y^3).
2. $g_x = y\,e^{xy}$ (chain rule, inner derivative $\partial(xy)/\partial x = y$; $\ln y$ is constant). $g_y = x\,e^{xy} + \frac{1}{y}$.
3. $h_x = yz$, $h_y = xz$, $h_z = xy + 2z$.
4. $f_x = 2xy^3$, so $f_{xy} = \partial_y(2xy^3) = 6xy^2$. $f_y = 3x^2y^2$, so $f_{yx} = \partial_x(3x^2y^2) = 6xy^2$. They are equal.
5. $\partial V/\partial h = lw = 3 \cdot 2 = 6$. Raising the height by a small amount Δh adds about $6\,\Delta h$ to the volume: a slab with the base area 6 and thickness Δh.

20.3. The gradient

20.3.1. The problem

A function of n variables has n partial derivatives. Each one describes the slope in one axis direction. Keeping track of them separately is clumsy. More importantly, the hiker doesn't only want to walk due east or due north. We want one object that captures all the slope information at a point and can answer questions about *any* direction. That object is the gradient.

20.3.2. A short refresher on vectors

The full story of vectors is in Chapter 26. For this chapter you need only the following.



Vector, length, unit vector

A **vector** in $\mathbb{R}^2$ is an ordered pair $\mathbf{v} = (v_1, v_2)$, pictured as an arrow from the origin to the point (v_1, v_2). It has a direction and a length. The same idea works with n components.

Its **length** (or **magnitude**, or **norm**) is $|\mathbf{v}| = \sqrt{v_1^2 + v_2^2}$ (Pythagoras; see Volume 1, Chapter 10).

A **unit vector** has length 1 and is used purely to indicate a direction. Any nonzero vector can be turned into one by dividing by its length: $\mathbf{u} = \mathbf{v}/|\mathbf{v}|$. For example, $(3, 4)$ has length 5, so $(0.6,\ 0.8)$ is the unit vector in that direction.

Dot product

The **dot product** of two vectors multiplies matching components and adds the results:

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 \ (+ \cdots + a_n b_n).$$

It also has a geometric meaning, which Chapter 26 derives from the law of cosines:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}|\,|\mathbf{b}| \cos\theta,$$

where θ is the angle between the arrows. So the dot product is largest when the vectors point the same way ($\cos 0 = 1$), zero when they are perpendicular ($\cos 90^\circ = 0$), and most negative when they point opposite ways.

20.3.3. Definition

Gradient and the symbol ∇

The **gradient** of $f(x_1, \ldots, x_n)$ is the vector of its partial derivatives:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ \frac{\partial f}{\partial x_2}, \ \ldots, \ \frac{\partial f}{\partial x_n} \right).$$

The symbol ∇ is called **nabla** (or "del"). Read $\nabla f(\mathbf{x})$ as "the gradient of f at the point $\mathbf{x}$". It is a vector with one component for each input, and it can be different at every point. In some texts it is also written **grad** f or $\frac{\partial f}{\partial \mathbf{x}}$.

For our running example $f(x, y) = x^3 + 2xy^2 - y$, we already have both partials:

$$\nabla f(x, y) = \big(3x^2 + 2y^2, \ 4xy - 1\big), \qquad \nabla f(1, 2) = (11, \ 7).$$

So far this is just packaging. The payoff is two geometric facts, which the next two sections prove:

1. **Direction:** ∇f points in the direction in which f increases fastest.
2. **Magnitude:** $|\nabla f|$ is that fastest rate of increase.

There is also a corollary: ∇f is perpendicular to the contour line through the point.

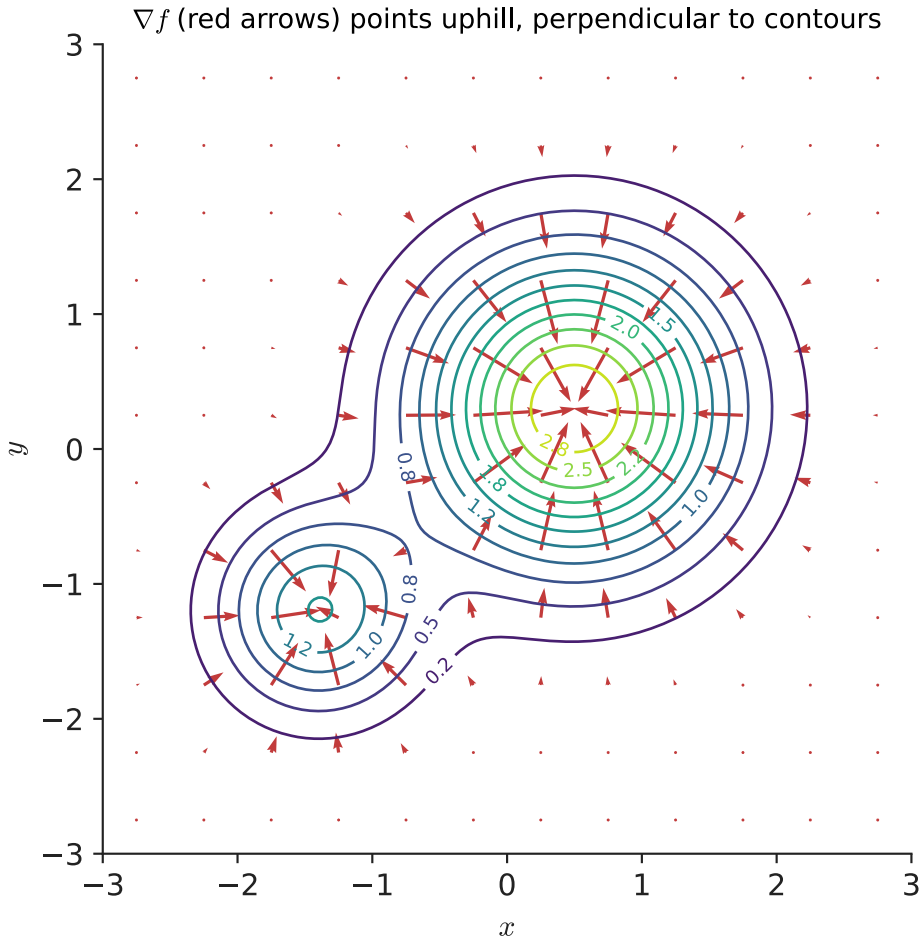


Figure 49. The gradient as a field of arrows. At each grid point the red arrow is ∇f. Every arrow points uphill and crosses the contours at right angles. Arrows are long where contours are bunched (steep) and short near the summits and on the flat plain (where $\nabla f \approx \mathbf{0}$).

The arrows shrink to nothing at the two peaks. At a summit there is no uphill direction, so the gradient is the zero vector. Remember that observation: finding where $\nabla f = \mathbf{0}$ is how we will find maxima and minima. *Try it:* Explore the gradient field .

20.3.4. Where you'd use this

- **Everyday: skiing and water.** On a hillside, the direction of $-\nabla h$ is the **fall line**: the way a ball rolls, water drains and a skier picks up speed fastest. Drainage engineers and hydrologists compute it from digital elevation maps to predict where floodwater will collect.
- **Physics and engineering: heat flow.** Heat flows down the temperature gradient: Fourier's law says the heat flux is $-k\nabla T$. If the temperature falls by 20 °C across a 10 cm brick wall, $|\nabla T| = 20/0.1 = 200$ °C per metre, and with brick's conductivity $k \approx 0.7$ W/(m·K) about 140 watts leak through every square metre. The same law with different letters governs diffusion of chemicals and the flow of groundwater; and in mechanics, force is minus the gradient of potential energy, $\mathbf{F} = -\nabla U$.
- **Computing: edge detection.** Treat a greyscale image as a function $I(x, y)$. Where brightness jumps from 50 to 200 across a boundary, the central-difference gradient is about $(200 - 50)/2 = 75$ per pixel pointing across the boundary, and nearly zero along it. Edge

detectors (Sobel, Canny) compute ∇I at every pixel and mark points where $|\nabla I|$ is large: the first stage of many computer-vision pipelines, from barcode scanners to lane detection in cars.

20.4. Tangent planes and linearisation

20.4.1. The problem

In one variable, the tangent line $y = f(a) + f'(a)(x - a)$ was the best straight-line approximation to a curve near a. Close to a, the curve and the line are almost the same. What is the multivariable version? Zoom in far enough on a smooth surface and it looks flat, like the ground under your feet on a round Earth. That flat thing is the **tangent plane**.

20.4.2. Why it works

Move from (a, b) to $(a + \Delta x,\ b + \Delta y)$. Do it in two legs: first east, then north.

- The east leg changes f by about $f_x(a, b)\,\Delta x$, since f_x is the slope in that direction.
- The north leg changes f by about $f_y\,\Delta y$. (Strictly, f_y is evaluated at the end of the first leg, but for small steps that is almost the same point.)

The two changes add:

$$f(a + \Delta x,\ b + \Delta y) \approx f(a, b) + f_x(a, b)\,\Delta x + f_y(a, b)\,\Delta y.$$

Written with $x = a + \Delta x$ and $y = b + \Delta y$, this is the **tangent plane** at (a, b):

$$z = L(x, y) = f(a, b) + f_x(a, b)(x - a) + f_y(a, b)(y - b).$$

In vector form, using the gradient and the dot product, it reads almost exactly like the one-variable tangent line:

$$f(\mathbf{a} + \Delta\mathbf{x}) \approx f(\mathbf{a}) + \nabla f(\mathbf{a}) \cdot \Delta\mathbf{x}.$$

This is the most important formula in the chapter. **The gradient is the thing you dot with a small step to get the change in f.** It works unchanged in a million dimensions.



Linearisation, tangent plane, differential

The **linearisation** (or **linear approximation**) of f at $\mathbf{a}$ is $L(\mathbf{x}) = f(\mathbf{a}) + \nabla f(\mathbf{a}) \cdot (\mathbf{x} - \mathbf{a})$. Its graph, for two inputs, is the **tangent plane**.

The **total differential** writes the same idea in terms of small changes: $df = f_x\,dx + f_y\,dy$.

20.4.3. Worked example

Find the tangent plane to $f(x, y) = x^3 + 2xy^2 - y$ at $(1, 2)$, and use it to estimate $f(1.1,\ 1.9)$. *Try it:* Tangent plane and its error .

$$
\begin{aligned}
f(1,2) &= 1 + 2(1)(4) - 2 = 7 && \text{(height of the point)} \\
f_x(1,2) &= 11, \quad f_y(1,2) = 7 && \text{(from the partial-derivative example)} \\
L(x,y) &= 7 + 11(x-1) + 7(y-2) && \text{(tangent-plane formula)} \\
L(1.1, 1.9) &= 7 + 11(0.1) + 7(-0.1) && (\Delta x = 0.1,\ \Delta y = -0.1) \\
&= 7 + 1.1 - 0.7 = 7.4.
\end{aligned}
$$

The true value is $f(1.1, 1.9) = 1.331 + 2(1.1)(3.61) - 1.9 = 1.331 + 7.942 - 1.9 = 7.373$. The estimate is off by 0.027, under 0.4%, for a step of 0.1 in each direction.

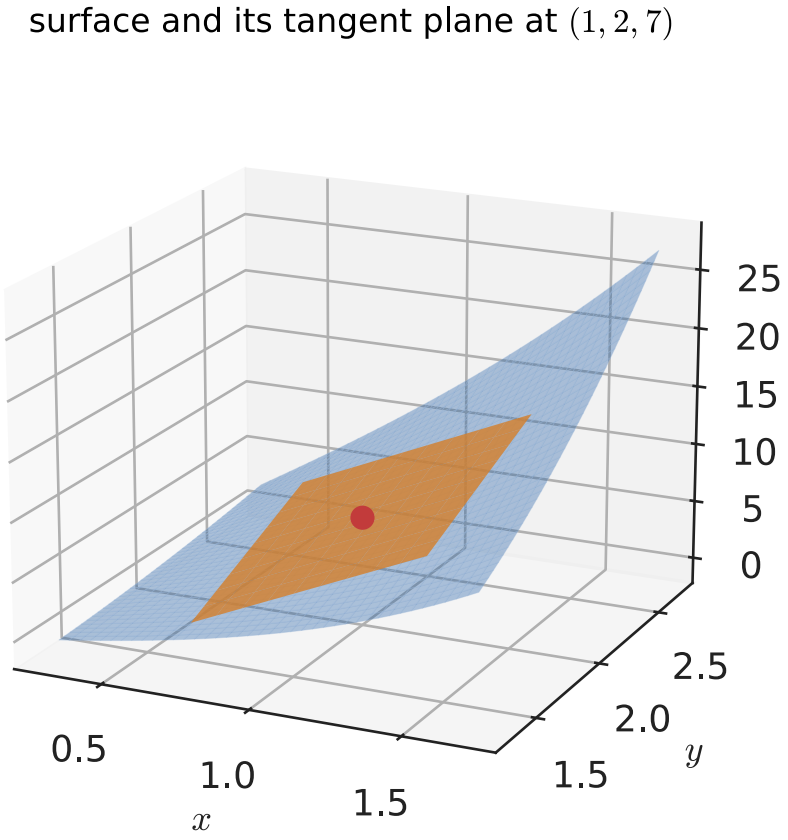


Figure 50. The tangent plane (orange) touches the surface (blue) at $(1, 2, 7)$ and matches its tilt in every direction.

20.4.4. Application: propagating measurement error

You measure a rectangle as $x = 20$ cm by $y = 10$ cm, and each measurement may be off by up to 0.1 cm. How uncertain is the area $A = xy$? *Try it:* Propagate the measurement error .

$$
\begin{aligned}
dA &= A_x\,dx + A_y\,dy = y\,dx + x\,dy && \text{(total differential)} \\
|dA| &\le 10(0.1) + 20(0.1) = 1 + 2 = 3 && \text{(worst case: both errors push the same way)}
\end{aligned}
$$

So $A = 200 \pm 3$ cm², about 1.5%. Notice that the error in the *short* side matters more, because it gets multiplied by the long side. Engineers and experimental scientists use this calculation every day under the name **propagation of uncertainty**.

Linearisation is the engine of optimisation

Almost every continuous optimisation algorithm works by building a local model of the function and stepping according to the model. Gradient descent uses the linear model $f(\mathbf{x}+\Delta\mathbf{x}) \approx f(\mathbf{x}) + \nabla f \cdot \Delta\mathbf{x}$. Newton's method adds a quadratic term (the Hessian, below). In automatic differentiation, a "forward-mode" pass computes exactly the directional change $\nabla f \cdot \Delta\mathbf{x}$, known in that field as a **Jacobian–vector product**.

20.5. Directional derivatives and the direction of steepest ascent

20.5.1. The problem

Our hiker at $(1, 2)$ on $f(x, y) = x^3 + 2xy^2 - y$ wants to walk in the direction of the unit vector $\mathbf{u} = (0.6,\ 0.8)$, roughly north-north-east. How fast does height change per unit distance walked? And of all possible directions, which climbs fastest?

20.5.2. Why it works: derive the directional derivative

Take a small step of length h in direction $\mathbf{u}$, so $\Delta\mathbf{x} = h\mathbf{u}$. Linearisation says:

$$f(\mathbf{a}+h\mathbf{u}) - f(\mathbf{a}) \approx \nabla f(\mathbf{a}) \cdot (h\mathbf{u}) \qquad \text{(linearisation with } \Delta\mathbf{x} = h\mathbf{u}\text{)}$$

$$\frac{f(\mathbf{a}+h\mathbf{u}) - f(\mathbf{a})}{h} \approx \nabla f(\mathbf{a}) \cdot \mathbf{u} \qquad \text{(divide by the distance walked, } h\text{)}$$

As $h \to 0$ the approximation becomes exact (for smooth f), which gives the **directional derivative**:

$$D_{\mathbf{u}} f(\mathbf{a}) = \lim_{h \to 0} \frac{f(\mathbf{a}+h\mathbf{u}) - f(\mathbf{a})}{h} = \nabla f(\mathbf{a}) \cdot \mathbf{u}.$$

The partial derivatives are special cases. With $\mathbf{u} = (1, 0)$ (due east), $D_{\mathbf{u}} f = f_x$. With $\mathbf{u} = (0, 1)$, $D_{\mathbf{u}} f = f_y$.



Directional derivative

The **directional derivative** $D_{\mathbf{u}} f(\mathbf{a})$ is the rate of change of f at $\mathbf{a}$ per unit distance moved in the direction of the **unit** vector $\mathbf{u}$. It equals $\nabla f(\mathbf{a}) \cdot \mathbf{u}$.

20.5.3. Why the gradient is the steepest direction

Now use the geometric form of the dot product. Let θ be the angle between ∇f and $\mathbf{u}$. Since $|\mathbf{u}| = 1$:

$$D_{\mathbf{u}} f = \nabla f \cdot \mathbf{u} = |\nabla f|\,|\mathbf{u}| \cos\theta = |\nabla f| \cos\theta.$$

Read off the consequences. $|\nabla f|$ is fixed at the point, so only $\cos\theta$ varies as you turn to face different directions.

- $\cos\theta$ is largest, equal to 1, when $\theta = 0$: when you face along ∇f. The rate of climb is then $|\nabla f|$. **The gradient points in the direction of steepest ascent, and its length is the steepest slope.**

- $\cos\theta = -1$ when $\theta = 180°$: facing along $-\nabla f$ gives the steepest *descent*, at rate $-|\nabla f|$. That is the direction gradient descent will use.
- $\cos\theta = 0$ when $\theta = 90°$: walking perpendicular to the gradient, height doesn't change at all, to first order. But a path along which height doesn't change is exactly a contour line. So **the gradient is perpendicular to the contour through the point**.

That is the whole argument, and it rests on one fact about the dot product. It also explains the contour rule from the start of the chapter: bunched contours mean the height changes a lot over a short distance, so $|\nabla f|$ is large.

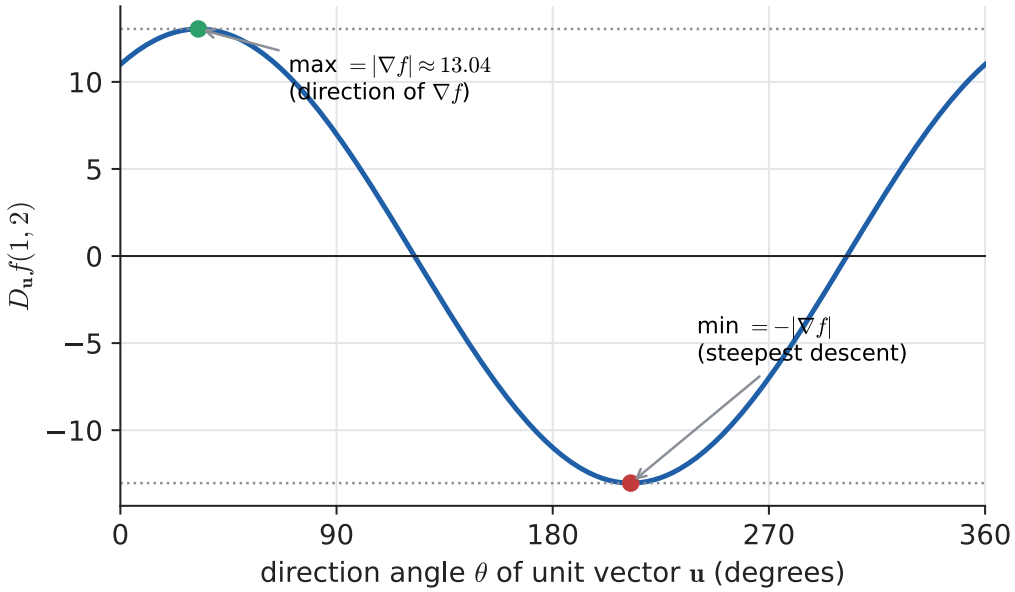


Figure 51. As the walking direction $\mathbf{u}$ turns through 360° at the point $(1, 2)$, the slope $D_{\mathbf{u}} f = |\nabla f| \cos\theta$ traces a cosine wave. Its peak is at the gradient's direction (about 32.5°), and the steepest descent is directly opposite.

20.5.4. The method

Example 22. Procedure — directional derivative and steepest directions

Input: a differentiable f, a point $\mathbf{a}$, and a direction vector $\mathbf{v}$ (any length).

Steps:

1. Compute the gradient $\nabla f(\mathbf{a})$ from the partial derivatives.
2. Normalise the direction: $\mathbf{u} = \mathbf{v}/|\mathbf{v}|$.
3. Directional derivative: $D_{\mathbf{u}} f(\mathbf{a}) = \nabla f(\mathbf{a}) \cdot \mathbf{u}$.
4. Steepest ascent direction: $\nabla f(\mathbf{a})/|\nabla f(\mathbf{a})|$, with rate $|\nabla f(\mathbf{a})|$.
5. Steepest descent direction: $-\nabla f(\mathbf{a})/|\nabla f(\mathbf{a})|$, with rate $-|\nabla f(\mathbf{a})|$.

Output: the rate of change in the requested direction, and the best and worst directions.

20.5.5. Worked example

At $(1, 2)$ on $f(x, y) = x^3 + 2xy^2 - y$, with direction $\mathbf{v} = (3, 4)$: *Try it:* Turn the direction dial .

$$\begin{aligned}
\nabla f(1, 2) &= (11,\ 7) && \text{(step 1, computed earlier)} \\
\mathbf{u} &= \frac{(3, 4)}{\sqrt{3^2 + 4^2}} = \frac{(3, 4)}{5} = (0.6,\ 0.8) && \text{(step 2)} \\
D_{\mathbf{u}} f(1, 2) &= 11(0.6) + 7(0.8) = 6.6 + 5.6 = 12.2 && \text{(step 3)} \\
|\nabla f(1, 2)| &= \sqrt{11^2 + 7^2} = \sqrt{170} \approx 13.04 && \text{(step 4: the best possible rate)} \\
\text{steepest ascent} &= \frac{(11, 7)}{\sqrt{170}} \approx (0.844,\ 0.537) && \text{(about 32.5° above east)}
\end{aligned}$$

So walking north-north-east climbs at 12.2 units of height per unit of distance. That is close to the best possible 13.04, because $(0.6, 0.8)$ points at about 53° and the gradient at about 32.5°, only 20° apart: $13.04 \cos 20.6° \approx 12.2$. Walking in direction $(-0.844, -0.537)$ descends fastest. Walking in direction $(-7, 11)/\sqrt{170}$, perpendicular to the gradient, keeps you on the contour $f = 7$.



The direction must be a unit vector

$\nabla f \cdot \mathbf{v}$ with an unnormalised $\mathbf{v}$ gives the wrong number. With $\mathbf{v} = (3, 4)$ above, you'd get $33 + 28 = 61$, five times too big, because $|\mathbf{v}| = 5$. Normalise first.

Also, "the gradient points uphill" is a *local* statement. It gives the steepest direction right here. It does not point at the summit. On a curved ridge, following the gradient traces a curving path, as the arrows in the gradient-field figure show.

20.5.6. Reading $\nabla f(\mathbf{x})$ fluently

This is the literacy goal from the preface. When you meet

$$\nabla f(\mathbf{x})$$

in a paper, unpack it in four layers:

1. **Symbols:** f is a scalar function of many variables. The bold $\mathbf{x}$ is the point (a vector of inputs) where we evaluate. ∇ means "the gradient of".
2. **Definition:** it is a vector of partial derivatives, $(\partial f/\partial x_1, \ldots, \partial f/\partial x_n)$, with one entry per input. Entry i says how sensitive f is to input i alone.
3. **Geometry:** it points in the direction of steepest increase of f, its length is that steepest rate, and it is perpendicular to the level set through $\mathbf{x}$.
4. **Use:** an optimisation algorithm uses it to decide which way to move. Step along $-\nabla f$ to decrease f (gradient descent) or along $+\nabla f$ to increase it.

If you can say those four sentences without looking, you have the main idea of this chapter.

Why "nabla"?

∇ is an upside-down Greek capital delta. The name "nabla" comes from the Greek for a Hebrew harp that has roughly that triangular shape. William Rowan Hamilton introduced the symbol (lying on its side) in the 1840s. The physicist Peter Guthrie Tait set it upright. The name was suggested to Tait by the biblical scholar William Robertson Smith, and James Clerk Maxwell used it, half in jest, in his letters. Physicists mostly call it "del".

Follow the arrow

Click a point on a contour map. The app draws ∇f there and a draggable unit vector $\mathbf{u}$. As you rotate $\mathbf{u}$, a live readout shows $D_{\mathbf{u}} f = |\nabla f| \cos\theta$ and a dot moves along the cosine wave. Confirm by hand that the maximum is along the gradient and the zeros are along the contour.

Try it: Open the Follow the arrow lab

US curriculum

Gradients and directional derivatives are core **Calculus III** topics (college, or post-AP in some high schools). The dot product that drives the argument appears in **Precalculus** (vectors unit) and in high-school physics.

20.5.7. Exercises

1. Compute $\nabla f(3, 1)$ for $f(x, y) = x^2 + 3y^2$, and give the unit vector in the direction of steepest descent there.
2. For $f(x, y) = x^2 y$ at $(2, 1)$, find the directional derivative towards the point $(5, 5)$, and the maximum rate of increase.
3. For $f(x, y) = x^2 + y^2$, verify that $\nabla f(3, 4)$ is perpendicular to the level curve through $(3, 4)$. (Hint: the level curve is a circle. Its tangent direction at $(3, 4)$ is $(-4, 3)$.)
4. Use linearisation to estimate $\sqrt{3.1^2 + 3.9^2}$ from the point $(3, 4)$, and compare with the true value.

Answers

1. $\nabla f = (2x, 6y)$, so $\nabla f(3, 1) = (6, 6)$, with length $\sqrt{72} = 6\sqrt{2}$. Steepest descent: $-(6, 6)/(6\sqrt{2}) = (-1/\sqrt{2}, -1/\sqrt{2}) \approx (-0.707, -0.707)$.
2. $\nabla f = (2xy, x^2) = (4, 4)$ at $(2, 1)$. Direction $(5, 5) - (2, 1) = (3, 4)$, so $\mathbf{u} = (0.6, 0.8)$. $D_{\mathbf{u}} f = 4(0.6) + 4(0.8) = 2.4 + 3.2 = 5.6$. Maximum rate $|\nabla f| = 4\sqrt{2} \approx 5.657$.
3. $\nabla f = (2x, 2y) = (6, 8)$. Dot with the tangent direction: $(6)(-4) + (8)(3) = -24 + 24 = 0$, so they are perpendicular. The gradient points radially outwards, straight away from the circle's centre.

4. $f = \sqrt{x^2 + y^2}$, $f(3, 4) = 5$, $f_x = x/f = 0.6$, $f_y = y/f = 0.8$.
   $L = 5 + 0.6(0.1) + 0.8(-0.1) = 5 + 0.06 - 0.08 = 4.98$. True value:
   $\sqrt{9.61 + 15.21} = \sqrt{24.82} \approx 4.98197$.

20.6. The multivariable chain rule

20.6.1. The problem

A drone flies a path $(x(t), y(t))$ through a room where the temperature is $T(x, y)$. Its sensor reads $T(x(t), y(t))$, a function of time alone. How fast does the reading change? It changes for two reasons: x is changing, and y is changing. We need a way to combine them.

20.6.2. Intuition and derivation

Over a short time Δt, the drone moves by $\Delta x \approx x'(t)\,\Delta t$ and $\Delta y \approx y'(t)\,\Delta t$. Linearisation says the temperature changes by

$$\Delta T \approx T_x\,\Delta x + T_y\,\Delta y \approx T_x\,x'(t)\,\Delta t + T_y\,y'(t)\,\Delta t.$$

Divide by Δt and let $\Delta t \to 0$:

$$\frac{dT}{dt} = \frac{\partial T}{\partial x}\frac{dx}{dt} + \frac{\partial T}{\partial y}\frac{dy}{dt} = \nabla T \cdot \mathbf{r}'(t),$$

where $\mathbf{r}'(t) = (x'(t), y'(t))$ is the drone's velocity. The single-variable chain rule "multiply the rates" becomes "**multiply along each path of dependence, then add over the paths**". A tree diagram makes the bookkeeping mechanical.

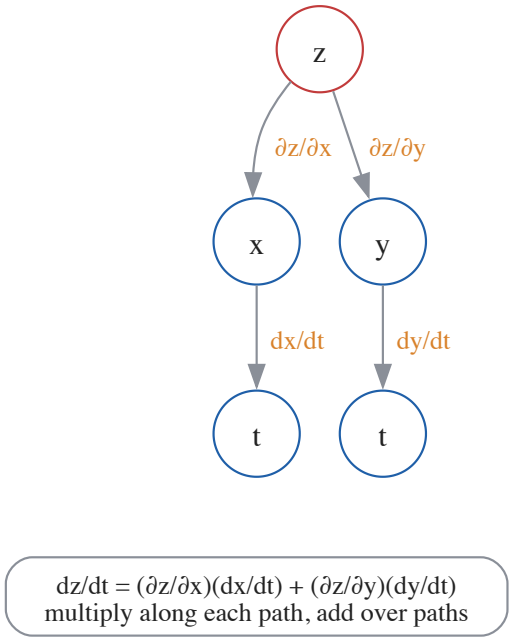


Figure 52. Dependency tree. z depends on x and y, each of which depends on t. Each edge carries a derivative. Multiply along every root-to-leaf path that ends at t, then add.

The same rule handles intermediate variables that depend on several inputs. If $z = f(x, y)$ with $x = x(s, t)$ and $y = y(s, t)$, then

$$\frac{\partial z}{\partial s} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial s}, \qquad \frac{\partial z}{\partial t} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial t}.$$

Example 23. Procedure — multivariable chain rule

Input: an outer function $z = f(x_1, \dots, x_n)$ whose inputs depend on further variables. The goal is the derivative of z with respect to one of those further variables, t.

Steps:

1. Draw the dependency tree from z down to every occurrence of t.
2. Label each edge with the corresponding (partial) derivative.
3. For each path from z to t, multiply the edge labels.
4. Add the products over all paths.
5. Substitute the current values of all variables.

Output: $\partial z/\partial t$ (or dz/dt if t is the only underlying variable).

20.6.3. Worked example

The temperature is $T(x, y) = x^2 + xy$ and the drone flies around the unit circle, $x = \cos t$, $y = \sin t$. Find dT/dt at $t = \pi/4$.

$$\begin{aligned}
T_x = 2x + y, \quad T_y = x && \text{(partials of the outer function)} \\
\frac{dx}{dt} = -\sin t, \quad \frac{dy}{dt} = \cos t && \text{(derivatives along the path)} \\
\frac{dT}{dt} = (2x + y)(-\sin t) + x\cos t && \text{(multiply along paths, add)}
\end{aligned}$$

At $t = \pi/4$, $x = y = \sin(\pi/4) = \cos(\pi/4) = \frac{\sqrt{2}}{2}$:

$$\begin{aligned}
\frac{dT}{dt} &= \left(2 \cdot \tfrac{\sqrt{2}}{2} + \tfrac{\sqrt{2}}{2}\right)\left(-\tfrac{\sqrt{2}}{2}\right) + \tfrac{\sqrt{2}}{2} \cdot \tfrac{\sqrt{2}}{2} \\
&= \left(\tfrac{3\sqrt{2}}{2}\right)\left(-\tfrac{\sqrt{2}}{2}\right) + \tfrac{2}{4} && (\sqrt{2} \cdot \sqrt{2} = 2) \\
&= -\tfrac{6}{4} + \tfrac{2}{4} = -1.
\end{aligned}$$

Check by the alternative method. Substitute first and use one-variable calculus. $T(t) = \cos^2 t + \cos t \sin t = \cos^2 t + \frac{1}{2}\sin 2t$. Then $T'(t) = -2\cos t \sin t + \cos 2t = -\sin 2t + \cos 2t$, and $T'(\pi/4) = -1 + 0 = -1$. The answers agree.

Choosing a method. For a small problem with explicit formulas, substituting first is often quicker. The chain-rule form wins when the functions are only known numerically, when there are many intermediate variables, or when you need the derivatives with respect to *many* inputs at once. That last case is what neural networks require.

Where you'd use this. Any sensor moving through a field measures a chain-rule derivative. A drone flying at velocity $(5, 3)$ m/s through air where $\nabla T = (-0.02, 0.01)$ °C per metre sees its temperature reading change at $\frac{dT}{dt} = T_x \frac{dx}{dt} + T_y \frac{dy}{dt} = (-0.02)(5) + (0.01)(3) = -0.07$ °C per second, about 4 °C a minute, even if the air itself is not changing at all. Weather balloons, ocean gliders and a phone's barometric altimeter all rely on this split between "the field changed" and "I moved through the field". In economics, the same formula says how a cost index moves when several prices drift at once.

Backpropagation is the multivariable chain rule

A neural network is a long composition of functions: inputs feed layers, layers feed layers, and at the end a single number, the **loss** L, measures how wrong the prediction was. Training needs ∇L with respect to every weight. **Backpropagation** is the chain rule applied to the dependency graph, evaluated from the loss backwards. Each node multiplies the derivative it receives from above by its own local derivative and passes the result down. Where paths merge, the contributions are added, exactly as in the tree rule.

Here is the smallest possible case: one "neuron" $p = wx + b$ with squared-error loss $L = (p - y)^2$. The dependency chain is $w \to p \to L$, so $\partial L/\partial w = \frac{\partial L}{\partial p} \cdot \frac{\partial p}{\partial w} = 2(p - y) \cdot x$.

PYTHON

```python
# One "neuron": prediction p = w*x + b, loss L = (p - y)^2
x, y = 2.0, 7.0          # one training example
w, b = 1.5, 0.5          # current parameters

p = w*x + b              # forward pass
L = (p - y)**2

dL_dp = 2*(p - y)        # backward pass: chain rule, outermost first
dL_dw = dL_dp * x        # because dp/dw = x
dL_db = dL_dp * 1        # because dp/db = 1
print("p =", p, " L =", L)
print("grad L =", (dL_dw, dL_db))

# gradient check with central differences
h = 1e-6
num_w = (((w+h)*x + b - y)**2 - ((w-h)*x + b - y)**2) / (2*h)
num_b = ((w*x + b + h - y)**2 - (w*x + b - h - y)**2) / (2*h)
print("numerical  =", (round(num_w, 6), round(num_b, 6)))
```

```
p = 3.5  L = 12.25
grad L = (-14.0, -7.0)
numerical  = (-14.0, -7.0)
```

Both components of the gradient are negative, so increasing w and b lowers the loss. That makes sense: the prediction 3.5 is below the target 7. The final lines are a **gradient check**, a standard debugging step whenever you hand-code derivatives. The whole of modern deep learning (PyTorch, JAX, TensorFlow) rests on doing this backward pass automatically for graphs with billions of nodes. Volume 3, Chapter 33 builds a small network this way.

20.6.4. The Jacobian, briefly

So far the outer function returned a single number. Many maps return several numbers: a robot arm maps joint angles (θ_1, θ_2) to a hand position (x, y). A graphics transform maps (u, v) texture coordinates to screen positions. For a map $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m$, put the gradients of the m output components in the rows of a matrix:



Jacobian matrix

The **Jacobian** of $\mathbf{F}(x, y) = \big(F_1(x, y), F_2(x, y)\big)$ is the matrix of all first partial derivatives:

$$J = \begin{bmatrix} \dfrac{\partial F_1}{\partial x} & \dfrac{\partial F_1}{\partial y} \\[2ex] \dfrac{\partial F_2}{\partial x} & \dfrac{\partial F_2}{\partial y} \end{bmatrix}.$$

Row i is ∇F_i. The Jacobian is the multi-output version of the derivative: small input changes $\Delta\mathbf{x}$ produce output changes $\Delta\mathbf{F} \approx J\,\Delta\mathbf{x}$. Matrix–vector multiplication is explained in Chapter 27.

Example: the polar-to-Cartesian map $x = r\cos\theta,\ y = r\sin\theta$ has Jacobian

$$J = \begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix}, \qquad \det J = r\cos^2\theta + r\sin^2\theta = r.$$

The determinant r is the area-scaling factor. That's why a small polar rectangle has area $r\,dr\,d\theta$, a fact we'll use for double integrals. In robotics, the Jacobian of the arm's kinematics converts joint speeds into hand velocity. Inverting it ("which joint speeds give the hand velocity I want?") is the core of real-time arm control. The arm is in trouble exactly where $\det J = 0$, a **singular configuration** such as a fully outstretched arm, which cannot move its hand any further outward.



Leibniz's notation pays off

The chain rule $\frac{dz}{dt} = \frac{\partial z}{\partial x}\frac{dx}{dt} + \cdots$ looks as if the ∂x and dx "cancel". They don't really, because a partial derivative is not a fraction, and in some thermodynamics identities naive cancellation gives the wrong sign. Still, the notation was designed to suggest the right answer most of the time, and it is one reason Leibniz's notation beat Newton's dots.

20.6.5. Exercises

1. Let $z = x^2 y$ with $x = t^2$ and $y = 3t$. Find dz/dt at $t = 1$ by the chain rule, then check by substituting first.
2. Draw the dependency tree for $w = f(x, y, z)$ with x, y, z each a function of s and t, and write the formula for $\partial w/\partial s$.

3. In the neuron example, compute $\partial L/\partial w$ when $x = 1$, $y = 0$, $w = 2$, $b = 1$. Which way should w move to reduce the loss?

▼ **Answers**

1. $z_x = 2xy$, $z_y = x^2$, $x' = 2t$, $y' = 3$. At $t = 1$: $x = 1, y = 3$.
   $dz/dt = (2 \cdot 1 \cdot 3)(2) + (1)(3) = 12 + 3 = 15$. Check: $z = t^4 \cdot 3t = 3t^5$, so $dz/dt = 15t^4 = 15$ at $t = 1$.
2. Three paths from w to s: $\frac{\partial w}{\partial s} = \frac{\partial w}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial w}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial w}{\partial z}\frac{\partial z}{\partial s}$.
3. $p = 2(1) + 1 = 3$, $\partial L/\partial p = 2(3 - 0) = 6$, $\partial L/\partial w = 6 \cdot x = 6$. The derivative is positive, so *decrease* w to reduce the loss.

20.7. Critical points and the Hessian

20.7.1. The problem

In Chapter 18 you found the maxima and minima of $f(x)$ by solving $f'(x) = 0$ and then checking the second derivative. What is the multivariable version, and what new possibilities appear when there is more than one direction?

20.7.2. Intuition

At the top of a smooth hill or the bottom of a smooth bowl, the ground is level in every direction: the tangent plane is horizontal. That means every directional derivative is zero, which happens exactly when $\nabla f = \mathbf{0}$.

But a level tangent plane does not guarantee a peak or a pit. Think of a horse saddle or a mountain pass. Along the path over the pass you are at a high point. Along the ridge that the pass cuts through you are at a low point. The tangent plane is horizontal, yet the point is neither a maximum nor a minimum. This **saddle point** has no counterpart worth mentioning in one variable, and it turns out to be extremely common in high dimensions.

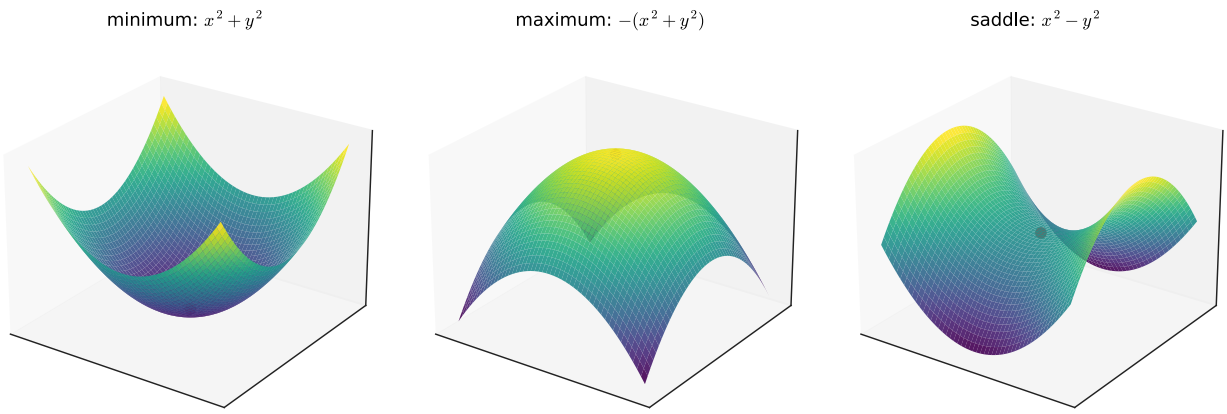


Figure 53. Three kinds of critical point. In each case $\nabla f = \mathbf{0}$ at the origin (red dot), but only the first two are extrema.

Critical point, saddle point, Hessian

A **critical point** (or **stationary point**) of f is a point where $\nabla f = \mathbf{0}$, i.e. every first partial derivative is zero.

A **saddle point** is a critical point that is neither a local maximum nor a local minimum: f rises in some directions and falls in others.

The **Hessian** is the matrix of second partial derivatives:

$$H = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}.$$

It is symmetric ($f_{xy} = f_{yx}$) and plays the role of the second derivative: it measures curvature.

20.7.3. Why it works: the second-derivative test

Near a critical point the linear term of the approximation vanishes (the gradient is zero), so the shape is decided by the next term, the quadratic one built from the Hessian. For two variables, the type of that quadratic is decided by one number, the **discriminant**

$$D = f_{xx} f_{yy} - f_{xy}^2 = \det H.$$

Roughly: if $D > 0$, the surface curves the same way in every direction (a bowl or a dome), and the sign of f_{xx} says which. If $D < 0$, it curves up in some directions and down in others, which is a saddle. For the general n-variable picture, the signs of the Hessian's *eigenvalues* decide: all positive means a minimum, all negative a maximum, mixed a saddle. That's one of the reasons Chapter 28 matters.

Example 24. Procedure — classifying critical points (two variables)

Input: a function $f(x, y)$ with continuous second partials.

Steps:

1. Compute f_x and f_y. Solve $f_x = 0$ and $f_y = 0$ simultaneously to get the critical points.
2. Compute f_{xx}, f_{yy}, f_{xy}.
3. At each critical point evaluate $D = f_{xx} f_{yy} - f_{xy}^2$.
4. $D > 0$ and $f_{xx} > 0$: local minimum. $D > 0$ and $f_{xx} < 0$: local maximum. $D < 0$: saddle point. $D = 0$: the test is inconclusive.

Output: each critical point with its classification.

20.7.4. Worked example

Classify the critical points of $f(x, y) = x^3 - 3x + y^2$. *Try it:* Classify these critical points .

$$
\begin{aligned}
f_x = 3x^2 - 3 = 0 \;&\Rightarrow\; x^2 = 1 \;\Rightarrow\; x = \pm 1 && \text{(step 1)} \\
f_y = 2y = 0 \;&\Rightarrow\; y = 0 \\
&\text{critical points: } (1, 0) \text{ and } (-1, 0) \\
f_{xx} = 6x, \quad f_{yy} = 2, \quad f_{xy} = 0 \;& && \text{(step 2)} \\
D(1, 0) = (6)(2) - 0^2 = 12 > 0,\; f_{xx} = 6 > 0 \;&\Rightarrow\; \text{local minimum} && \text{(steps 3–4)} \\
D(-1, 0) = (-6)(2) - 0^2 = -12 < 0 \;&\Rightarrow\; \text{saddle point}
\end{aligned}
$$

The local minimum value is $f(1, 0) = 1 - 3 + 0 = -2$. Is it the *global* minimum? No. Along the x-axis, $f(x, 0) = x^3 - 3x \to -\infty$ as $x \to -\infty$, so the function has no lowest value at all. The second-derivative test is purely local.



Local is not global, and D = 0 tells you nothing

- A local minimum is only the lowest point in its neighbourhood. To find a global minimum you must also compare all the critical points and check the boundary of the region or the behaviour at infinity, just as in one variable.
- When $D = 0$ the test says nothing. $x^4 + y^4$, $-(x^4 + y^4)$ and $x^4 - y^4$ all have $D = 0$ at the origin, which is a minimum, a maximum and a saddle respectively.
- Solving $\nabla f = \mathbf{0}$ by hand only works for tidy functions. For a neural network's loss with millions of variables, the system of equations is hopeless to solve directly. That is why we need gradient descent.

Try it: See D = 0 fail .

20.7.5. Where you'd use this

- **Science: fitting a line to data.** "Least squares" chooses the slope m and intercept c that minimise $S(m, c) = \sum_i (y_i - mx_i - c)^2$. Setting $\nabla S = \mathbf{0}$ gives two linear equations (the **normal equations**). For the points $(1, 2)$, $(2, 3)$, $(3, 5)$ they give $m = 1.5$, $c = \frac{1}{3}$, and the Hessian test ($S_{mm} = 28$, $S_{cc} = 6$, $S_{mc} = 12$, $D = 24 > 0$) confirms a minimum. Every spreadsheet trendline and every linear-regression library starts here (Volume 3, Chapter 31).
- **Chemistry.** A molecule's energy is a function of all its atoms' positions. Stable molecules sit at minima; the **transition state** of a reaction, the highest point on the easiest path between reactant and product, is a saddle point, and chemistry software locates it with exactly the Hessian test above.
- **Computing.** Machine-learning losses are classified the same way, as the sidebar below explains: the gradient finds the critical points, the Hessian says what kind they are.



> *Saddles dominate high dimensions*
>
> For a critical point to be a minimum in n dimensions, the surface must curve upward in *every* one of n independent directions. If each direction were a coin flip, the chance would be 2^{-n}. In a million dimensions, almost every critical point is a saddle. Research in the 2010s suggested that this, rather than "bad local minima", is the main geometric obstacle in training deep networks, and that noisy gradient methods are good at slipping off saddles.

20.8. Gradient descent

20.8.1. The problem

You want the minimum of a function $f(\mathbf{x})$: the loss of a model, the energy of a molecule, the cost of a design. You can compute f and ∇f at any point, but solving $\nabla f = \mathbf{0}$ symbolically is impossible, because there are too many variables and the equations are too tangled. What can you do?

20.8.2. Intuition: walking downhill in fog

You are on a mountainside in thick fog and want to reach the valley floor. You can't see the valley, but you can feel the slope under your feet. A sensible strategy: feel which way is steepest downhill, take a step that way, and repeat. The gradient gives you exactly that local information. Steepest descent is the direction $-\nabla f$, and that gives the iteration

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k).$$

Here $\mathbf{x}_k$ is the position after k steps and η (Greek "eta") is the step-size multiplier.

20.8.3. Why it works

Linearisation tells us how much one step changes f. With $\Delta\mathbf{x} = -\eta\nabla f$:

$$f(\mathbf{x}_{k+1}) \approx f(\mathbf{x}_k) + \nabla f \cdot (-\eta \nabla f) = f(\mathbf{x}_k) - \eta |\nabla f(\mathbf{x}_k)|^2.$$

Since $|\nabla f|^2 > 0$ whenever we're not at a critical point, a small enough positive η is guaranteed to decrease f. The words "small enough" are where all the practical difficulty lives, because the linear approximation is only trustworthy for small steps. Notice also that the step length is $\eta|\nabla f|$: steps are automatically long on steep ground and short on gentle ground, so the method slows down as it nears a minimum.



Learning rate, iterate, convergence

The **learning rate** (or **step size**) η is the positive number multiplying the gradient in each update. Machine learning calls it the learning rate. Numerical analysis calls it the step size and often writes α.

Each $\mathbf{x}_k$ is an **iterate**. The method **converges** if the iterates approach a limit (ideally a minimum), and **diverges** if they run off, typically growing without bound.

20.8.4. The method

Example 25. Procedure — gradient descent

Input: a differentiable function f and a way to compute ∇f; a starting point $\mathbf{x}_0$; a learning rate $\eta > 0$; a tolerance ε; an iteration cap K.

Steps:

1. Set $k = 0$.
2. Compute $\mathbf{g} = \nabla f(\mathbf{x}_k)$.
3. If $|\mathbf{g}| < \varepsilon$, stop: you are (nearly) at a critical point.
4. Update $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta\,\mathbf{g}$.
5. If the iterates are blowing up, or f is rising, stop and reduce η.
6. Increase k. If $k < K$, go to step 2.

Output: $\mathbf{x}_k$, an approximate local minimiser, and the number of steps taken.

20.8.5. Worked example: three steps by hand

Minimise $f(x, y) = x^2 + 3y^2$ (the elliptical bowl from the first section) starting at $\mathbf{x}_0 = (3, 1)$ with $\eta = 0.1$. The gradient is $\nabla f = (2x,\ 6y)$. *Try it:* Step down the bowl .

$$\mathbf{x}_0 = (3,\ 1), \qquad f(\mathbf{x}_0) = 9 + 3 = 12$$

$$\nabla f(\mathbf{x}_0) = (6,\ 6)$$

$$\mathbf{x}_1 = (3 - 0.1 \cdot 6,\ \ 1 - 0.1 \cdot 6) = (2.4,\ 0.4), \qquad f(\mathbf{x}_1) = 5.76 + 0.48 = 6.24$$

$$\nabla f(\mathbf{x}_1) = (4.8,\ 2.4)$$

$$\mathbf{x}_2 = (2.4 - 0.48,\ \ 0.4 - 0.24) = (1.92,\ 0.16), \qquad f(\mathbf{x}_2) = 3.6864 + 0.0768 = 3.7632$$

$$\nabla f(\mathbf{x}_2) = (3.84,\ 0.96)$$

$$\mathbf{x}_3 = (1.92 - 0.384,\ \ 0.16 - 0.096) = (1.536,\ 0.064), \qquad f(\mathbf{x}_3) = 2.359296 + 0.012288 \approx 2.3716$$

The loss falls 12 → 6.24 → 3.76 → 2.37. Look at the pattern in the coordinates. Each step multiplies x by 0.8 and y by 0.4. That's no accident:

$$x_{k+1} = x_k - \eta(2x_k) = (1 - 2\eta)\, x_k,$$
$$y_{k+1} = y_k - \eta(6y_k) = (1 - 6\eta)\, y_k.$$

With $\eta = 0.1$ the factors are 0.8 and 0.4. The y coordinate, in the steep direction, collapses quickly, while x, in the shallow direction, shrinks only by 20% per step. That slow direction sets the overall pace.

20.8.6. Implementation and experiment

PYTHON

```
def gradient_descent(grad, x0, eta, tol=1e-8, max_iter=10_000):
    x = list(x0)
    for k in range(max_iter):
        g = grad(x)
        if sum(gi*gi for gi in g) ** 0.5 < tol:   # gradient ~ 0: stop
            return x, k
        x = [xi - eta*gi for xi, gi in zip(x, g)]  # step downhill
        if any(abs(xi) > 1e12 for xi in x):
            raise OverflowError(f"diverged at iteration {k}")
    return x, max_iter

grad = lambda p: [2*p[0], 6*p[1]]      # gradient of f(x, y) = x^2 + 3y^2

for eta in [0.01, 0.1, 0.25, 0.3, 0.34, 0.5]:
    try:
        x, k = gradient_descent(grad, [3.0, 1.0], eta)
        print(f"eta={eta:<5} converged in {k:5d} steps to "
              f"({x[0]:.1e}, {x[1]:.1e})")
    except OverflowError as e:
        print(f"eta={eta:<5} {e}")
```

```
eta=0.01  converged in  1001 steps to (4.9e-09, 1.3e-27)
eta=0.1   converged in    91 steps to (4.6e-09, 6.1e-37)
eta=0.25  converged in    30 steps to (2.8e-09, 9.3e-10)
eta=0.3   converged in    91 steps to (1.8e-36, -1.5e-09)
eta=0.34  diverged at iteration 704
eta=0.5   diverged at iteration 39
```

The multiplier formulas explain every line of this output.

- **Too small ($\eta = 0.01$):** the factors are 0.98 and 0.94. Every step is safe, but x shrinks by only 2% per step, so it takes over a thousand steps. The method **crawls**.
- **Good ($\eta = 0.1$):** factors 0.8 and 0.4. Steady progress.
- **Best ($\eta = 0.25$):** factors 0.5 and -0.5. Both coordinates shrink at the same rate (the y coordinate flips sign each step but halves in size). This balances the slow and fast directions, and it is the fastest constant step for this bowl.

- **Getting large ($\eta = 0.3$):** factors 0.4 and -0.8. Now y is the slow coordinate, and it flips sign every step. The path **zig-zags** across the valley.
- **Too large ($\eta = 0.34$):** the y factor is $1 - 6(0.34) = -1.04$. Its magnitude exceeds 1, so y grows by 4% each step while bouncing side to side, and the iteration slowly **diverges**. At $\eta = 0.5$ the factor is -2 and the blow-up is fast.

So convergence requires $|1 - 2\eta| < 1$ *and* $|1 - 6\eta| < 1$, i.e. $0 < \eta < 1/3$. **The steepest direction sets the upper limit on the learning rate, and the shallowest direction sets the speed.** The 2 and 6 are the eigenvalues of this bowl's Hessian, $H = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix}$, and their ratio, the **condition number** $6/2 = 3$, measures how elongated the bowl is. Long thin valleys (large condition number) are slow for gradient descent no matter how you tune η. Chapter 28 and Chapter 32 develop that idea.

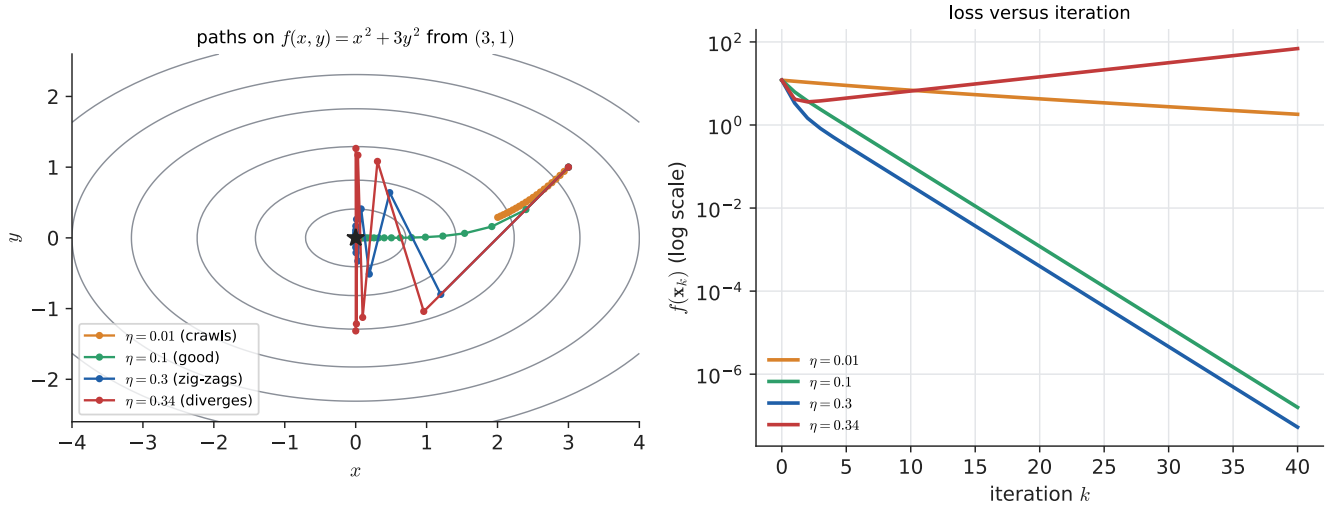


Figure 54. Left: gradient-descent paths from $(3, 1)$ on the bowl $x^2 + 3y^2$. Small η creeps along, moderate η glides in, $\eta = 0.3$ zig-zags across the narrow direction, and $\eta = 0.34$ bounces outward. Right: loss against iteration on a log scale. The straight lines mean the error shrinks geometrically, by a constant factor per step.

20.8.7. Alternative approaches and choosing a method

- **Solve $\nabla f = 0$ exactly.** Best when it's possible: quadratic functions, a handful of variables, tidy formulas. Linear regression with squared loss can be solved this way via the "normal equations" (Volume 3, Chapter 33). *Try it:* Try learning rates yourself .
- **Newton's method.** Use the Hessian to build a quadratic model and jump to its minimum: $\mathbf{x}_{k+1} = \mathbf{x}_k - H^{-1}\nabla f$. On our bowl it lands on the exact minimum in one step, because the bowl *is* quadratic. It converges very quickly near a minimum, but each step needs the $n \times n$ Hessian and a linear solve, which is impossible for a million variables (a 10^{12}-entry matrix).
- **Gradient descent and its relatives.** Cheap per step (one gradient), scales to huge n. The relatives (stochastic gradient descent, momentum, Adam) dominate machine learning and appear in Volume 3, Chapter 33 .
- **Derivative-free methods** (grid search, Nelder–Mead, evolutionary algorithms). Use these when no gradient is available or f is noisy or discontinuous. They are much slower in many dimensions.

Rule of thumb: few variables and a formula, solve analytically. Tens to thousands of smooth variables, use Newton-type or quasi-Newton methods (BFGS) from a library. Millions of variables, or data arriving in batches, use gradient descent variants.



How gradient descent fails

- **Learning rate too large:** oscillation, then divergence (loss shoots to infinity or NaN). The first fix to try for an exploding loss is to divide η by 10.
- **Learning rate too small:** painfully slow progress that can look like convergence. A flat loss curve does not prove you're at the minimum.
- **Ill-conditioning:** in long narrow valleys the gradient points mostly across the valley rather than along it, which produces zig-zags. Rescaling the inputs (feature normalisation) often helps a great deal.
- **Local minima and saddles:** gradient descent finds *a* point where $\nabla f \approx \mathbf{0}$, not necessarily the best one. Different starting points can lead to different answers.
- **Units matter:** the gradient changes if you rescale a variable (say metres to millimetres), so the "steepest" direction depends on your choice of units.

Loss surfaces in machine learning

Training a model means minimising a **loss function** $L(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the vector of all parameters (weights). For linear regression with squared error, L is a perfect convex bowl with a single minimum. For a deep network it is a vast, non-convex landscape with saddles, plateaus and many minima. Practitioners plot **loss curves** (loss against training step, like the right panel above) to diagnose the learning rate: a curve that explodes means η is too large, and a curve that barely moves means it is too small. Learning-rate **schedules** (start large, decay over time) and adaptive methods like Adam, which effectively choose a different η for each coordinate, are engineering responses to exactly the failure modes in the experiment above.



Cauchy, 1847

Gradient descent was proposed by Augustin-Louis Cauchy in 1847, in a short note on solving the systems of equations that arose in computing planetary orbits. He suggested reducing a positive function by stepping against its partial derivatives. A method designed to save astronomers some arithmetic now trains nearly every machine-learning model in the world.

Tune the learning rate

Pick a surface (round bowl, elongated bowl, a banana-shaped Rosenbrock valley, a two-minimum landscape). Click to choose a starting point and drag a slider for η. The app animates the descent path on the contour plot and draws the loss curve beside it. Find the largest η that still converges on the elongated bowl, and check it against the prediction $2/\lambda_{\max}$. On the two-minimum landscape, colour each starting point by which minimum it reaches.

Try it: Colour the two basins · Open the learning-rate lab · Descend the Rosenbrock valley

US curriculum

Gradient descent is not taught in US high schools. It appears in college courses on numerical methods, optimisation and machine learning. The one-variable idea (step toward lower values using the derivative) can be explored by a strong **AP Calculus** student. The ingredients, derivatives and iteration, are all in AP Calculus AB.

20.8.8. Exercises

1. Do two steps of gradient descent by hand on $f(x, y) = (x - 1)^2 + 2(y + 2)^2$ from $(0, 0)$ with $\eta = 0.1$.
2. For the same function, what is the largest constant learning rate for which gradient descent converges from any start?
3. Explain in a sentence why the loss curve in the right-hand figure is a straight line on a log scale for $\eta = 0.1$.
4. A colleague's training loss goes 2.3, 2.1, 2.6, 4.9, 31, NaN. What is the most likely cause, and what should they try?

▼ **Answers**

1. $\nabla f = (2(x - 1),\ 4(y + 2))$. At $(0, 0)$: $(-2, 8)$, so $\mathbf{x}_1 = (0 + 0.2,\ 0 - 0.8) = (0.2, -0.8)$. At $(0.2, -0.8)$: $\nabla f = (-1.6,\ 4.8)$, so $\mathbf{x}_2 = (0.2 + 0.16,\ -0.8 - 0.48) = (0.36, -1.28)$. Both coordinates head towards the minimum $(1, -2)$.
2. The error factors are $1 - 2\eta$ for x and $1 - 4\eta$ for y. We need $|1 - 4\eta| < 1$, so $\eta < 0.5$. (The Hessian is $\mathbf{diag}(2, 4)$ and $2/\lambda_{\max} = 0.5$.)
3. Each step multiplies each coordinate by a fixed factor (0.8 or 0.4), so the loss is multiplied by roughly a fixed factor per step. Geometric decay $c\,r^k$ has $\log(c\,r^k) = \log c + k \log r$, which is linear in k.
4. The learning rate is too large: the loss first falls, then oscillates and blows up. Reduce η (try one tenth), and consider gradient clipping or a warm-up schedule.

20.9. Constrained optimisation: Lagrange multipliers

20.9.1. The problem

You have 8 metres of fencing to build a rectangular pen against a long wall. The wall forms one side, so the fencing covers one side of length x parallel to the wall and two sides of length y. Maximise the area $A = xy$ subject to $x + 2y = 8$.

This is optimisation *with a constraint*. You can't wander freely over the (x, y) plane, only along the line $x + 2y = 8$. Real problems are full of constraints: budgets, capacities, physical laws, probabilities that must sum to 1.

20.9.2. Intuition: the contour that just touches

Draw the contours of the objective $f = xy$ and the constraint line. Walk along the line. As you go, you cross contours of higher and higher area, until you reach a point where the line only *touches* a contour without crossing it. Beyond it, you'd cross back to lower contours. At that touching point the constraint curve and the contour are tangent.

Two curves that are tangent at a point share a tangent line there, so their perpendiculars are parallel. The perpendicular to a contour of f is ∇f. The constraint curve is itself a contour of the function $g(x, y) = x + 2y$, so its perpendicular is ∇g. Therefore, at the optimum,

$$\nabla f = \lambda \nabla g \quad \text{for some number } \lambda, \qquad g(x, y) = c.$$

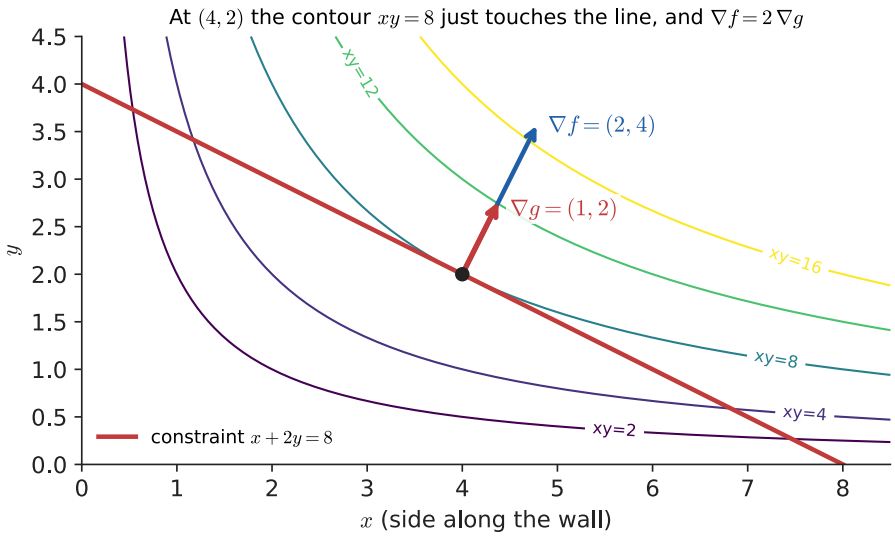


Figure 55. Maximising xy along the line $x + 2y = 8$. The best point $(4, 2)$ is where the line just touches a contour, here $xy = 8$. There, ∇f and ∇g are parallel.



Lagrange multiplier

To optimise $f(\mathbf{x})$ subject to a constraint $g(\mathbf{x}) = c$, look for points where $\nabla f = \lambda \nabla g$ and $g = c$. The number λ (Greek "lambda") is the **Lagrange multiplier**. It measures how fast the optimal value of f changes as the constraint level c is relaxed: $\lambda = df^*/dc$. Economists call it the **shadow price** of the constraint.

Example 26. Procedure — Lagrange multipliers (one equality constraint)

Input: an objective $f(x, y)$ and a constraint $g(x, y) = c$.

Steps:

1. Compute ∇f and ∇g.
2. Write the equations $f_x = \lambda g_x$, $f_y = \lambda g_y$, $g(x, y) = c$: three equations in three unknowns x, y, λ.
3. Solve them. It often helps to express x and y in terms of λ, then substitute into the constraint.
4. Evaluate f at every solution (and at the ends of the constraint curve, if it has any). The largest value is the maximum, the smallest the minimum.

Output: the constrained optimum, its location, and λ.

20.9.3. Worked example

Maximise $f(x, y) = xy$ subject to $g(x, y) = x + 2y = 8$. *Try it:* Watch the gradients align .

$$
\begin{aligned}
\nabla f = (y,\ x), \qquad \nabla g = (1,\ 2) \qquad & \text{(step 1)} \\
y = \lambda \cdot 1, \qquad x = \lambda \cdot 2 \qquad & \text{(step 2: components of } \nabla f = \lambda \nabla g) \\
2\lambda + 2(\lambda) = 8 \qquad & \text{(substitute into } x + 2y = 8) \\
4\lambda = 8 \ \Rightarrow \ \lambda = 2 \qquad & \text{(step 3)} \\
x = 4, \qquad y = 2, \qquad A = xy = 8\ \mathrm{m}^2 \qquad & \text{(step 4)}
\end{aligned}
$$

The ends of the feasible segment ($x = 0$ or $y = 0$) give area 0, so $(4, 2)$ is the maximum. The side along the wall is twice each of the other sides.

Alternative: substitution. From the constraint, $x = 8 - 2y$, so $A(y) = (8 - 2y)y = 8y - 2y^2$. Then $A'(y) = 8 - 4y = 0$ gives $y = 2$, $x = 4$. Same answer, with ordinary one-variable calculus.

Checking the meaning of λ. With c metres of fence instead of 8, the same steps give $\lambda = c/4$, $x = c/2$, $y = c/4$, so the best area is $A^* = c^2/8$. Then $dA^*/dc = c/4$, which equals 2 at $c = 8$. That is exactly λ. One extra metre of fence buys about 2 extra square metres of pen.

Choosing a method. Substitution is easier when you can solve the constraint for one variable cleanly. Lagrange multipliers win when you can't (constraints like $x^2 + xy + y^4 = 1$), when there are many variables and constraints, when symmetry makes the λ equations easy, or when the value of λ itself is what you want to know.



Lagrange finds candidates, not answers

- The condition $\nabla f = \lambda \nabla g$ identifies *candidates*: maxima, minima and saddle-like points all satisfy it. Always evaluate f at each candidate and compare.
- It needs $\nabla g \neq \mathbf{0}$ at the optimum. Points where the constraint curve has a sharp corner or cusp must be checked separately.
- Inequality constraints ($g \leq c$) need the more general **Karush–Kuhn–Tucker (KKT)** conditions. Either the optimum is inside the region (and $\nabla f = \mathbf{0}$ there) or it sits on the boundary (and Lagrange applies with a sign condition on λ).

Applications of constrained optimisation

- **Economics:** maximise utility subject to a budget. With £40 to spend on coffees at £2 and cakes at £4, and "utility" $U = xy$, the condition $(y, x) = \lambda(2, 4)$ with $2x + 4y = 40$ gives $\lambda = 2.5$, so 10 coffees and 5 cakes. The multiplier is the **marginal utility of money**: one more pound of budget buys about 2.5 more units of utility. *Try it:* Coffee and cake, interactively .
- **Physics:** a bead on a wire minimises energy subject to staying on the wire. λ turns out to be the constraint force.
- **Statistics and information theory:** the maximum-entropy distribution with a given mean is found with Lagrange multipliers, and it is the exponential distribution (Volume 3, Chapter 30 , Volume 3, Chapter 33).
- **Machine learning:** support vector machines are formulated as constrained optimisation, and their famous "support vectors" are the data points whose multipliers are nonzero.

US curriculum

Lagrange multipliers belong to **Calculus III**. US high-school students meet constrained optimisation in two forms: substitution word problems in **AP Calculus** ("maximise the area of a pen with fixed fencing") and **linear programming** in **Algebra II**, which optimises a linear objective over a polygon of linear constraints.

20.9.4. Exercises

1. Minimise $x^2 + y^2$ subject to $x + y = 4$, using Lagrange multipliers. Interpret geometrically.
2. Find the maximum and minimum of $f(x, y) = x + y$ on the circle $x^2 + y^2 = 2$. *Try it:* Check this exercise .

▼ **Answers**

1. $\nabla f = (2x, 2y) = \lambda(1, 1)$, so $x = y = \lambda/2$. The constraint gives $\lambda = 4$, so $x = y = 2$ and the minimum is 8. Geometrically, $(2, 2)$ is the point of the line closest to the origin, at distance $\sqrt{8} = 2\sqrt{2}$. The smallest circle centred at the origin that touches the line touches it there.

2. $(1,1) = \lambda(2x, 2y)$, so $x = y = 1/(2\lambda)$. The constraint gives $2/(4\lambda^2) = 2$, so $\lambda = \pm\tfrac{1}{2}$. $\lambda = \tfrac{1}{2}$ gives $(1,1)$ with $f = 2$ (maximum). $\lambda = -\tfrac{1}{2}$ gives $(-1,-1)$ with $f = -2$ (minimum).

20.10. Double integrals

20.10.1. The problem

In Chapter 19, $\int_a^b f(x)\,dx$ added up infinitely many thin strips to find the area under a curve. Now suppose $f(x, y)$ is the depth of snow over a rectangular field. How much snow is there? You need the **volume** under the surface $z = f(x, y)$ over a region of the plane.

20.10.2. Intuition and why it works

Chop the region into small rectangles of width Δx and depth Δy. Over each one, stand a column whose height is the value of f at some sample point inside it. Each column has volume $f \cdot \Delta x\,\Delta y$. Add them all up and refine the grid. The limit of these sums is the **double integral**:

$$\iint_R f(x,y)\,dA = \lim_{\Delta x,\,\Delta y \to 0} \sum_{\text{cells}} f(x_i, y_j)\,\Delta x\,\Delta y.$$

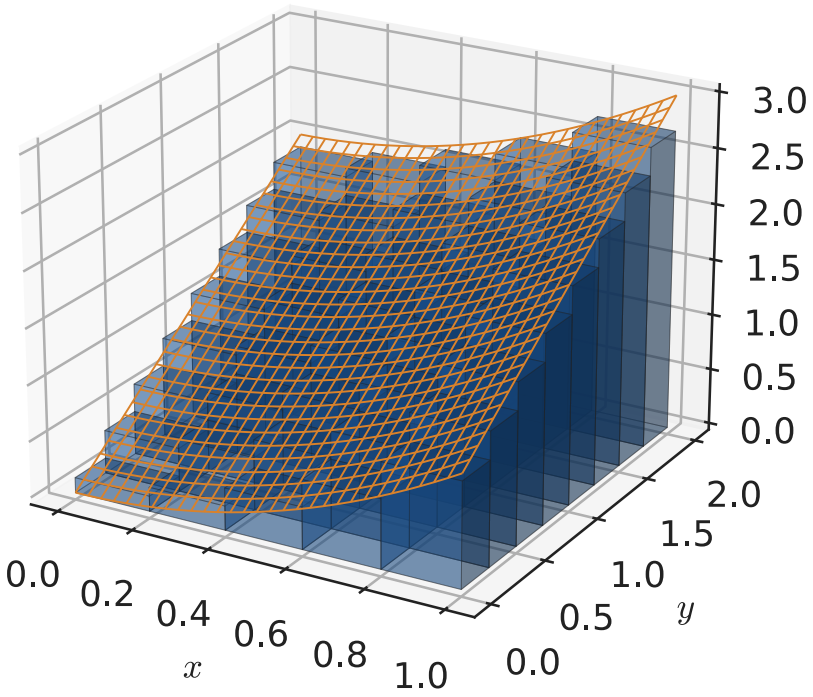


Figure 56. Approximating the volume under $z = x^2 + y$ by 40 columns, each with height taken at the midpoint of its base cell.

How do you *compute* it? Add the columns one row at a time. Fix x and integrate over y. That gives the area of one vertical slice through the solid, a function of x. Then integrate those slice areas over x, exactly as you found volumes by slicing in Chapter 19. This is an **iterated integral**, and **Fubini's**

theorem guarantees that for continuous f on a rectangle, the order doesn't matter:

$$\iint_{[a,b]\times[c,d]} f\,dA = \int_a^b \left(\int_c^d f(x,y)\,dy\right)dx = \int_c^d \left(\int_a^b f(x,y)\,dx\right)dy.$$



Double integral, iterated integral, dA

The **double integral** $\iint_R f\,dA$ is the signed volume between the surface $z = f(x,y)$ and the region R of the plane. dA stands for a tiny element of area, $dx\,dy$ in Cartesian coordinates and $r\,dr\,d\theta$ in polar coordinates (the r is the Jacobian determinant from earlier).

An **iterated integral** evaluates it as two ordinary integrals, inner first. While doing the inner integral, the outer variable is held constant, exactly as with partial derivatives.

$[a,b]\times[c,d]$ (read "a to b cross c to d") is the rectangle of points with $a \le x \le b$ and $c \le y \le d$.

20.10.3. Worked example

Find $\iint_R (x^2+y)\,dA$ over $R = [0,1]\times[0,2]$. *Try it:* Refine the columns .

Order 1: y inside. Hold x fixed:

$$\int_0^2 (x^2+y)\,dy = \left[x^2 y + \tfrac{1}{2}y^2\right]_{y=0}^{y=2} \qquad (x^2 \text{ is a constant here})$$
$$= (2x^2+2) - 0 = 2x^2+2$$
$$\int_0^1 (2x^2+2)\,dx = \left[\tfrac{2}{3}x^3 + 2x\right]_0^1 = \tfrac{2}{3}+2 = \tfrac{8}{3}.$$

Order 2: x inside. Hold y fixed:

$$\int_0^1 (x^2+y)\,dx = \left[\tfrac{1}{3}x^3 + xy\right]_{x=0}^{x=1} = \tfrac{1}{3}+y$$
$$\int_0^2 \left(\tfrac{1}{3}+y\right)dy = \left[\tfrac{1}{3}y + \tfrac{1}{2}y^2\right]_0^2 = \tfrac{2}{3}+2 = \tfrac{8}{3}.$$

Both orders give $8/3 \approx 2.667$, as Fubini promised. Dividing by the area of R (which is 2) gives the **average value** of f over the region, $4/3$.

20.10.4. Numerical check

The column picture translates directly into code, as the midpoint rule in two dimensions:

```python
import numpy as np

def double_integral_midpoint(f, ax, bx, ay, by, n):
    dx, dy = (bx - ax)/n, (by - ay)/n
    xs = ax + dx*(np.arange(n) + 0.5)      # midpoints in x
    ys = ay + dy*(np.arange(n) + 0.5)      # midpoints in y
    X, Y = np.meshgrid(xs, ys)
    return f(X, Y).sum() * dx * dy         # sum of column volumes

f = lambda x, y: x**2 + y
for n in [5, 50, 500]:
    print(n, double_integral_midpoint(f, 0, 1, 0, 2, n))
print("exact", 8/3)
```

```
5 2.660000000000001
50 2.6666000000000003
500 2.666666
exact 2.6666666666666665
```

Each tenfold refinement cuts the error by a factor of 100, the $O(h^2)$ behaviour of the midpoint rule. But in d dimensions an n-per-side grid needs n^d evaluations. That "curse of dimensionality" is why high-dimensional integrals (in physics, finance and Bayesian statistics) are done by **Monte Carlo** sampling instead, which Chapter 32 covers.



Non-rectangular regions

Over a region that isn't a rectangle, the inner limits depend on the outer variable. Over the triangle $0 \le y \le x \le 1$, for instance, you write $\int_0^1 \int_0^x f\,dy\,dx$. Swapping the order means re-describing the region ($\int_0^1 \int_y^1 f\,dx\,dy$), not just swapping the symbols. Always sketch the region first.

Applications of double integrals

- **Probability:** for two random variables with joint density $p(x, y)$, probabilities are double integrals, $P((X, Y) \in R) = \iint_R p\,dA$ (Volume 3, Chapter 30).
- **Physics:** the mass of a plate with variable density, the centre of mass, the moment of inertia.
- **Environment:** hydrologists find the total volume of rain falling on a catchment by integrating rainfall depth over its area. If the depth across a 3 km by 2 km catchment is $r(x, y) = 10 + 2x$ mm (wetter towards the east), the total is $\int_0^3 \int_0^2 (10 + 2x)\,dy\,dx = 2(30 + 9) = 78$ mm·km², and since 1 mm over 1 km² is 1000 m³, that is 78,000 m³ of water heading for the river.
- **Graphics:** rendering integrates incoming light over a hemisphere of directions at each pixel. Path tracers estimate those integrals by Monte Carlo.

US curriculum

Double and triple integrals are **Calculus III** topics. AP Calculus covers "volumes by slicing" (cross-sections, discs, washers), which is the same idea with the slice areas supplied by geometry rather than by an inner integral.

20.10.5. Exercises

1. Evaluate $\int_0^2 \int_0^3 (2x + y)\,dy\,dx$ in both orders, and give the average value of $2x + y$ over the rectangle.
2. Set up, but do not evaluate, the integral of $f(x, y)$ over the triangle with vertices $(0, 0)$, $(2, 0)$, $(0, 2)$.

▼ **Answers**

1. y inside: $\int_0^3 (2x + y)\,dy = 6x + \frac{9}{2}$, then $\int_0^2 (6x + 4.5)\,dx = 12 + 9 = 21$. x inside: $\int_0^2 (2x + y)\,dx = 4 + 2y$, then $\int_0^3 (4 + 2y)\,dy = 12 + 9 = 21$. The area is 6, so the average value is $21/6 = 3.5$.
2. The hypotenuse is $x + y = 2$. With y inside: $\int_0^2 \int_0^{2-x} f(x, y)\,dy\,dx$.

20.11. Putting it together

Here is the chapter on one page, as a chain of ideas:

1. A function of several variables is a landscape. Picture it as a surface, contours or a heat map.
2. A **partial derivative** is the slope of a slice, with every other input frozen.
3. The **gradient** ∇f bundles the partials into a vector.
4. **Linearisation:** $f(\mathbf{a} + \Delta\mathbf{x}) \approx f(\mathbf{a}) + \nabla f \cdot \Delta\mathbf{x}$. Everything else follows from this.
5. **Directional derivative** $D_{\mathbf{u}} f = \nabla f \cdot \mathbf{u} = |\nabla f| \cos\theta$. So ∇f is the steepest-ascent direction, $|\nabla f|$ the steepest slope, and ∇f is perpendicular to the contours.
6. **Chain rule:** multiply along dependency paths and add, which is backpropagation in miniature.
7. **Critical points** satisfy $\nabla f = \mathbf{0}$. The Hessian classifies them as minimum, maximum or saddle.
8. **Gradient descent** steps along $-\nabla f$. The learning rate must be below about $2/\lambda_{\max}$ or it diverges, and if it is too small the method crawls.
9. **Lagrange multipliers:** at a constrained optimum, $\nabla f \parallel \nabla g$.
10. **Double integrals** compute volume by iterated slicing.

20.12. Interactive tools

Browser tools on the companion website that go with this chapter. Each example link in the chapter opens one of them with that example already set up.

- Surfaces, gradients and gradient descent

20.13. Mastery check

1. Unpack $\nabla f(\mathbf{x})$ in four sentences: symbols, definition, geometry, use.
2. Let $f(x, y) = x^2 y + y^3$. Compute $\nabla f(1, 2)$.
3. For the same f, find the directional derivative at $(1, 2)$ in the direction of $(3, 4)$, and the maximum possible rate of increase there.
4. Use the linearisation at $(1, 2)$ to estimate $f(1.02,\ 1.97)$.
5. Find and classify the critical point(s) of $g(x, y) = x^2 - 4xy + y^2$.
6. Gradient descent on $h(x, y) = 4x^2 + y^2$: what is the largest learning rate that converges, and what happens with $\eta = 0.3$?
7. Maximise xy subject to $2x + 2y = 20$ with a Lagrange multiplier, and state λ.
8. Evaluate $\iint_{[0,1]\times[0,1]} xy \, dA$.

Decision rule: All correct without hesitation → move on to Volume 3, Chapter 21 , or jump ahead to Chapter 26 to put the dot product on solid ground. Stuck on 1–4 → reread Section 20.3, Section 20.4 and Section 20.5. Stuck on 6 → reread Section 20.8 and rerun the Python experiment with your own η values. Stuck on more → work the section exercises first.

▼ **Mastery check answers**

1. **Symbols:** $\nabla f(\mathbf{x})$ is the gradient of the scalar function f, evaluated at the point $\mathbf{x} \in \mathbb{R}^n$. **Definition:** the vector of partial derivatives $(\partial f/\partial x_1, \ldots, \partial f/\partial x_n)$. **Geometry:** it points in the direction of steepest increase, its length is that steepest rate, and it is perpendicular to the level set through $\mathbf{x}$. **Use:** optimisation algorithms step along $-\nabla f$ to decrease f.
2. $f_x = 2xy = 4$, $f_y = x^2 + 3y^2 = 1 + 12 = 13$. $\nabla f(1, 2) = (4, 13)$.
3. $\mathbf{u} = (0.6, 0.8)$. $D_{\mathbf{u}} f = 4(0.6) + 13(0.8) = 2.4 + 10.4 = 12.8$. Maximum rate $|\nabla f| = \sqrt{16 + 169} = \sqrt{185} \approx 13.60$.
4. $f(1, 2) = 2 + 8 = 10$. $L = 10 + 4(0.02) + 13(-0.03) = 10 + 0.08 - 0.39 = 9.69$. (True value ≈ 9.6950.)
5. $g_x = 2x - 4y = 0$ and $g_y = -4x + 2y = 0$. The first gives $x = 2y$. Substituting into the second gives $-8y + 2y = 0$, so $y = 0$ and $x = 0$. Only critical point: $(0, 0)$. $g_{xx} = 2$, $g_{yy} = 2$, $g_{xy} = -4$, so $D = 4 - 16 = -12 < 0$: a saddle.
6. $\nabla h = (8x, 2y)$, with factors $1 - 8\eta$ and $1 - 2\eta$. Convergence needs $|1 - 8\eta| < 1$, i.e. $\eta < 1/4$. At $\eta = 0.3$ the x factor is $1 - 2.4 = -1.4$, so x flips sign and grows by 40% per step: divergence.

7. $(y, x) = \lambda(2, 2)$, so $x = y = 2\lambda$. The constraint $8\lambda = 20$ gives $\lambda = 2.5$, so $x = y = 5$ and the area is 25. (Check: with perimeter c, the best area is $c^2/16$, and $d/dc = c/8 = 2.5$ at $c = 20$.)
8. $\int_0^1 \int_0^1 xy \, dy \, dx = \int_0^1 \frac{x}{2} \, dx = \frac{1}{4}$.

Part VII: Linear Algebra

Chapter 26. Vectors

A number like 5 tells you how much. Plenty of quantities need more than that: a displacement of 5 km is useless unless you know which way, and a force of 5 newtons pushes in some direction. Vectors are the mathematics of "how much, and which way". This chapter rebuilds them from two complementary viewpoints: arrows you can draw, and lists of numbers you can compute with. We add and scale vectors, measure their length, and meet the single most useful operation in applied linear algebra, the dot product, which turns out to measure angle, projection, work done, brightness of a pixel, and the relevance of a document to a search query. We finish by stretching the humble $(3, 4)$ *into the 768-dimensional embeddings that power modern semantic search, and by noting where three-dimensional intuition stops being trustworthy.*

Where this chapter sits

You need: coordinates and distance in the plane (Volume 1, Chapter 10), Pythagoras and basic geometry (Volume 1, Chapter 9), sine, cosine and the unit circle (Volume 1, Chapter 13), and solving a pair of linear equations (Volume 1, Chapter 6). No calculus is needed anywhere in Part VII.

This chapter unlocks: matrices as transformations of vectors (Chapter 27), eigenvectors and principal component analysis (Chapter 28), gradients as vectors of partial derivatives (Chapter 20), and the vector-heavy parts of statistics and machine learning (Volume 3, Chapter 31 , Volume 3, Chapter 33).

Why this chapter matters. Vectors are how both physics and software describe the world. A sat-nav adds displacement vectors, a game engine adds velocity vectors sixty times a second, a phone's step counter measures the length of an acceleration vector, and every search box and recommendation feed ranks results by the angle between vectors. The handful of operations in this chapter (add, scale, measure, dot) is the vocabulary all of them share.

26.1. Quantities with direction

26.1.1. The problem

You leave a campsite and walk 3 km due east, then 4 km due north. How far are you from camp, and in which direction?

You already know how to answer this. Draw the two legs as a right angle, use Pythagoras for the straight-line distance, $\sqrt{3^2 + 4^2} = 5$ km, and use trigonometry for the direction, $\tan^{-1}(4/3) \approx 53.1°$ north of east. But notice what you had to keep track of. Neither "3" nor "4" alone described a leg of the walk; each was a distance **in a direction**. And the answer, too, is a distance in a direction.

Physics is full of such quantities. If you took Mechanics at A-level, you spent a year with them: displacement, velocity, acceleration, force, momentum. All of them need a size **and** a direction. Two forces of 10 N acting on a box might cancel completely (opposite directions), add to 20 N (same direction), or combine to about 14.1 N (at right angles). The number 10 alone can't tell you which.

So we need a mathematical object that carries direction as well as size, and rules for combining such objects that match what happens in the world. That object is the **vector**.

26.1.2. Intuition: an arrow, or a list of numbers

There are two ways to picture a vector, and the whole of linear algebra gets easier once you can switch between them without effort.

The arrow view. A vector is an arrow: it has a length and points in a direction. Crucially, it has **no fixed position**. The displacement "3 km east, 4 km north" is the same displacement whether you start at the campsite or at the car park. Slide the arrow anywhere without rotating or stretching it and it's still the same vector.

The list view. Put the arrow's tail at the origin of a coordinate system. Its tip lands at some point, say $(3, 4)$. Those two numbers record exactly how far the arrow goes in each axis direction, and they determine the arrow completely. So a vector **is** the list $(3, 4)$.

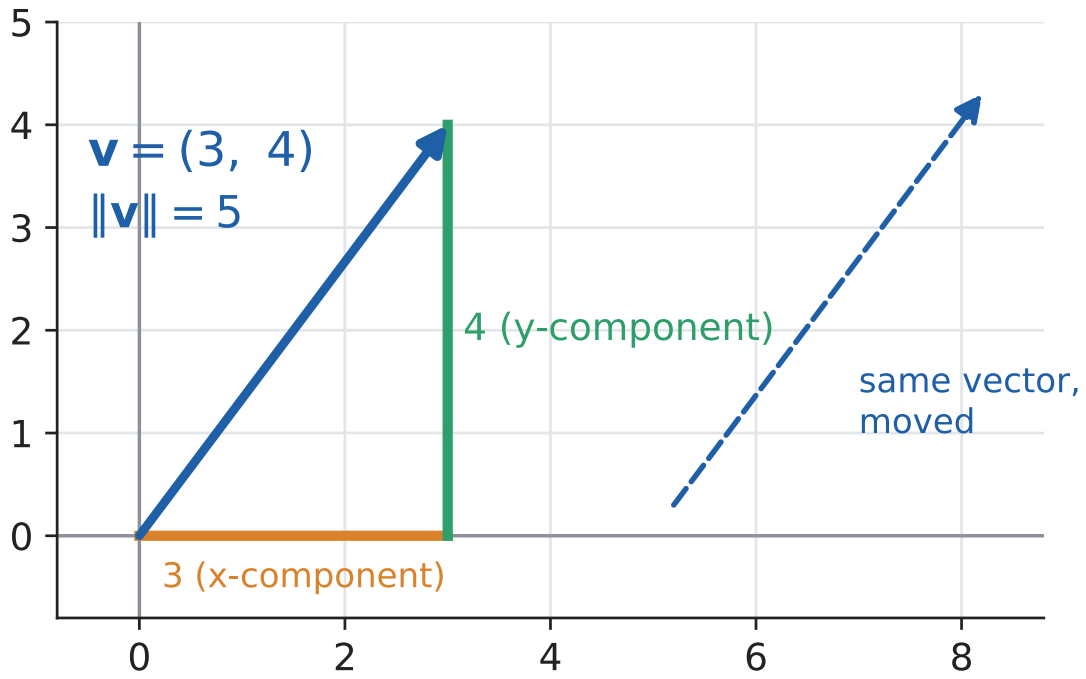


Figure 57. One vector, two views. The arrow from the origin to $(3, 4)$ is described completely by its components 3 and 4. Moving the arrow (dashed) doesn't change the vector.

The arrow view gives you geometry: you can **see** addition, angles and projections. The list view gives you computation: you can add, multiply and store vectors in an array. The list view also generalises effortlessly. An arrow in 768 dimensions is impossible to draw, but a list of 768 numbers is just an array, and every rule in this chapter applies to it unchanged.

Scalar and vector

A **scalar** is an ordinary single number: a size with no direction (mass 4 kg, temperature 20 °C, speed 30 m/s). A **vector** is a quantity with both **magnitude** (size) and **direction** (displacement 30 m north-east, velocity 30 m/s due south). In the list view, a vector in n dimensions is an ordered list of n real numbers. The word "scalar" comes from the fact that multiplying a vector by a number **scales** it.

Component

The **components** of a vector are the numbers in its list, one per axis. The vector $(3, 4)$ has **x-component** 3 and **y-component** 4. In general the i-th component of $\mathbf{v}$ is written v_i, so $\mathbf{v} = (v_1, v_2, \ldots, v_n)$. The set of all lists of n real numbers is written $\mathbb{R}^n$ ("R-n"), so $(3, 4) \in \mathbb{R}^2$ means "(3, 4) is a vector in two-dimensional real space".

26.1.3. Vocabulary and notation

Vector notation varies between textbooks, countries and fields. You'll meet all of these, and they mean the same thing.

Vector notation

- **Bold lower-case** in print: $\mathbf{v}$, $\mathbf{a}$, $\mathbf{F}$ (force).
- **An arrow or an underline** by hand, since you can't write in bold: $\vec{v}$ or $\underline{v}$. UK A-level Mechanics used the underline.
- **A column vector**, the standard in linear algebra:

$$\mathbf{v} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}.$$

- **A row or tuple**: $(3, 4)$ or $\langle 3, 4 \rangle$ (the angle brackets are common in US calculus books, to distinguish a vector from a point).
- **Unit-vector form**: $3\mathbf{i} + 4\mathbf{j}$, where $\mathbf{i}$ and $\mathbf{j}$ are the arrows of length 1 along the x- and y-axes (and $\mathbf{k}$ along the z-axis in 3D). This is the form you'll remember from Mechanics.
- **An arrow between points**: $\overrightarrow{AB}$ is the vector from point A to point B.

In this book, vectors are bold. We write them as tuples in running text to save space and as columns inside calculations, especially once matrices arrive in Chapter 27.



Points versus vectors

The point $(3, 4)$ and the vector $(3, 4)$ are written identically but are different kinds of thing. A point is a **location**; a vector is a **displacement**. The point $(3, 4)$ is "where the vector $(3, 4)$ lands if its tail is at the origin". The difference matters in practice: subtracting two points gives a vector (the displacement from one to the other), and adding a vector to a point gives a new point. Adding two **points** has no physical meaning: "London plus Paris" is not a place. Graphics programmers meet this distinction head-on, and it's exactly what **homogeneous coordinates** in Chapter 27 encode.



Why "vector"?

"Vector" is Latin for **carrier**, from *vehere*, to carry (the same root as "vehicle"). Astronomers used it for the imaginary line carrying a planet around the sun. The modern mathematical sense came from William Rowan Hamilton in the 1840s, as part of his quaternions (four-dimensional numbers he famously carved into Dublin's Broom Bridge). Quaternions were clumsy for everyday physics, and in the 1880s Josiah Willard Gibbs in America and Oliver Heaviside in England independently stripped them down to the vector algebra used today: the dot product, the cross product, and the bold-letter notation. Biologists still use the older sense: a mosquito is a "vector" because it **carries** malaria.

US curriculum

Vectors in two dimensions (magnitude, direction, components, addition, scalar multiplication) appear in **Precalculus** (Common Core standards N-VM.1–5, marked "+" as advanced). Many students first meet them seriously in **AP Physics 1** or **AP Physics C**. Three-dimensional vectors, the dot product and the cross product are standard in **Calculus III** (multivariable calculus) and in a first college **Linear Algebra** course.

26.2. Adding and scaling vectors

26.2.1. The problem

A light aircraft points due north and flies at 200 km/h through the air. A wind blows from the west at 50 km/h, so the air itself is moving east. What is the plane's actual velocity over the ground?

The plane's motion is the result of two motions happening at once. Each is a vector, and we need a rule for combining them.

26.2.2. Intuition: tip to tail

Think about displacements first, because they are the easiest to picture. If you walk along vector $\mathbf{a}$, then along vector $\mathbf{b}$, your total displacement is the single arrow from where you started to where you ended. Draw $\mathbf{b}$ starting at the tip of $\mathbf{a}$, then draw an arrow from the tail of $\mathbf{a}$ to the tip of $\mathbf{b}$. That arrow

is $\mathbf{a} + \mathbf{b}$. This is the **tip-to-tail rule**, also called the **triangle law**.

If you do the walks in the other order, $\mathbf{b}$ first, you trace the other two sides of a parallelogram and end up in the same place. So $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$, and the sum is the diagonal of the parallelogram the two vectors span. That's the **parallelogram law**, and it's how physicists add forces acting at a single point.

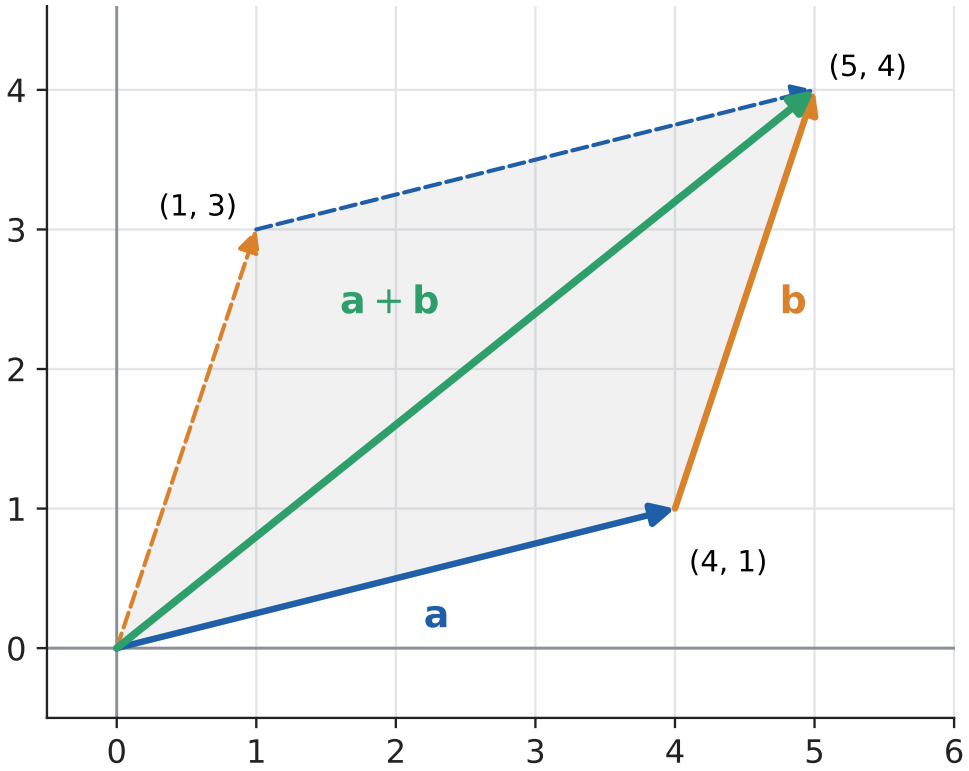


Figure 58. Adding $\mathbf{a} = (4, 1)$ and $\mathbf{b} = (1, 3)$. Tip-to-tail along either pair of sides of the parallelogram reaches the same point, $(5, 4)$.

26.2.3. Why it works: components add independently

Look at the x-coordinates in the diagram. Walking along $\mathbf{a}$ moves you 4 units right; then $\mathbf{b}$ moves you 1 more. Total: 5. The y-coordinates are independent of this: 1 up, then 3 more, total 4. Horizontal and vertical motions don't interfere with each other, so you add them separately:

$$\begin{bmatrix} 4 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 + 1 \\ 1 + 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}.$$

That is the whole rule, in any number of dimensions: **add component by component**. The arrow picture and the list calculation always agree, which is why both views are legitimate.

26.2.4. Scaling: stretch, shrink, flip

Walking along $\mathbf{v}$ twice is $\mathbf{v} + \mathbf{v}$, which in components is every entry doubled. We write it $2\mathbf{v}$. More generally, for any scalar c, the **scalar multiple** $c\mathbf{v}$ multiplies every component by c:

$$c \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} c\,v_1 \\ c\,v_2 \end{bmatrix}.$$

Geometrically, $c\mathbf{v}$ points along the same line as $\mathbf{v}$ and is $|c|$ times as long. If $c > 1$ it stretches, if $0 < c < 1$ it shrinks, and if $c < 0$ it also flips to point the opposite way. $0\mathbf{v}$ is the **zero vector** $\mathbf{0} = (0, 0)$, the only vector with no direction.

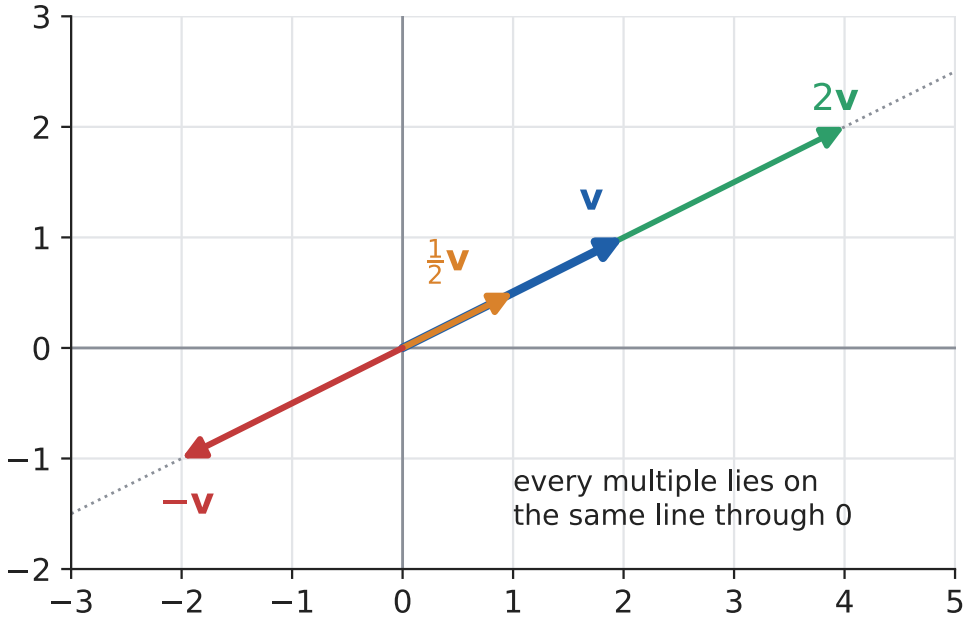


Figure 59. Scalar multiples of $\mathbf{v} = (2, 1)$. All of them lie on one line through the origin.

Subtraction follows: $\mathbf{a} - \mathbf{b} = \mathbf{a} + (-1)\mathbf{b}$. Its geometric meaning is worth memorising: $\mathbf{a} - \mathbf{b}$ is the arrow **from the tip of $\mathbf{b}$ to the tip of $\mathbf{a}$**. (Check: start at $\mathbf{b}$, add $\mathbf{a} - \mathbf{b}$, arrive at $\mathbf{a}$.) This is why "the vector from point P to point Q" is $Q - P$.



Vector arithmetic laws

For vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ in $\mathbb{R}^n$ and scalars s, t:

- $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$ (**commutative**)
- $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$ (**associative**)
- $\mathbf{a} + \mathbf{0} = \mathbf{a}$ and $\mathbf{a} + (-\mathbf{a}) = \mathbf{0}$
- $s(\mathbf{a} + \mathbf{b}) = s\mathbf{a} + s\mathbf{b}$ and $(s + t)\mathbf{a} = s\mathbf{a} + t\mathbf{a}$ (**distributive**)

Each follows immediately from the matching law for ordinary numbers, applied one component at a time. Any collection of objects obeying these rules is called a **vector space**; that abstraction is why polynomials, audio signals and images can all be treated as vectors.

Example 27. Procedure — adding and scaling vectors

Input: vectors $\mathbf{a} = (a_1, \dots, a_n)$, $\mathbf{b} = (b_1, \dots, b_n)$ of the same dimension; scalars s, t.

Steps:

1. Check the dimensions match. If they don't, the sum is undefined; stop.
2. Multiply every component of $\mathbf{a}$ by s and every component of $\mathbf{b}$ by t.
3. Add the results position by position: the i-th component of the answer is $sa_i + tb_i$.

Output: the vector $s\mathbf{a} + t\mathbf{b} = (sa_1 + tb_1, \dots, sa_n + tb_n)$.

26.2.5. Worked example: the aircraft in a crosswind

Set up axes with x pointing east and y pointing north, units km/h.

- Air velocity (where the plane points): $\mathbf{p} = (0, 200)$.
- Wind velocity (blowing **from** the west means moving **towards** the east): $\mathbf{w} = (50, 0)$.

Ground velocity is the sum:

$$\mathbf{g} = \mathbf{p} + \mathbf{w} = \begin{bmatrix} 0 + 50 \\ 200 + 0 \end{bmatrix} = \begin{bmatrix} 50 \\ 200 \end{bmatrix}.$$

To turn this back into "speed and direction", use Pythagoras and the tangent:

$$\begin{aligned} \text{ground speed} &= \sqrt{50^2 + 200^2} && \text{(length of the arrow)} \\ &= \sqrt{2500 + 40\,000} = \sqrt{42\,500} \approx 206.2 \text{ km/h}, \\ \text{drift angle} &= \tan^{-1}\!\left(\frac{50}{200}\right) && \text{(east of north)} \\ &\approx 14.0^\circ. \end{aligned}$$

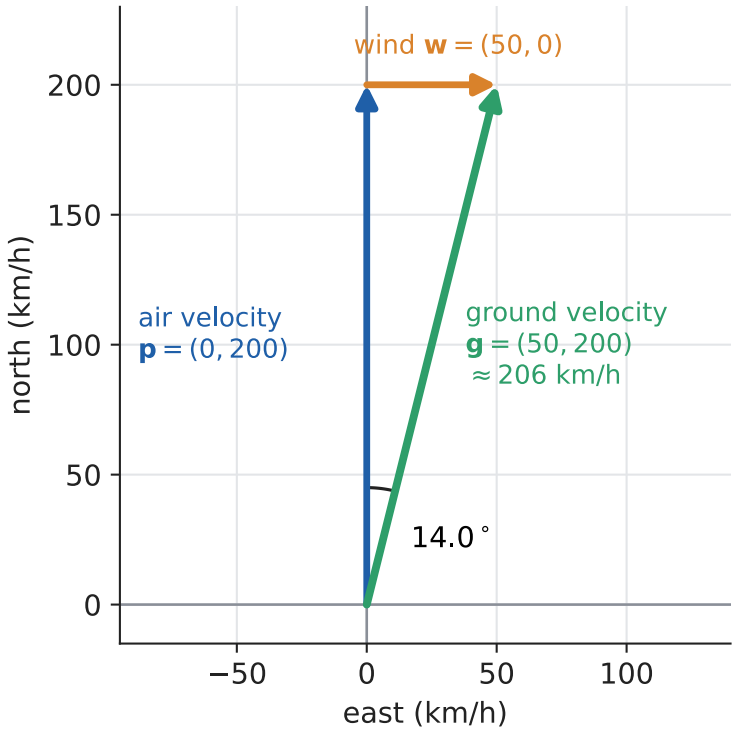


Figure 60. The crosswind as vector addition. The plane points north at 200 km/h (blue); the wind adds 50 km/h east (orange); the plane actually moves along the sum (green), 14.0° east of north at about 206 km/h.

The plane travels slightly faster than its airspeed and drifts 14° off course. A pilot who wants to track due north must point slightly **west** so that the wind's push cancels; finding that heading is a nice exercise below.

26.2.6. Worked example: three forces in equilibrium

A ring is held still by three ropes. Two of them pull with forces $\mathbf{F}_1 = (3, 4)$ N and $\mathbf{F}_2 = (-5, 2)$ N. What force must the third rope supply?

"Held still" (in **equilibrium**) means the forces sum to zero, Newton's first law. So

$$\begin{aligned} \mathbf{F}_1 + \mathbf{F}_2 + \mathbf{F}_3 &= \mathbf{0} \\ \mathbf{F}_3 &= -(\mathbf{F}_1 + \mathbf{F}_2) && \text{(subtract from both sides)} \\ &= -\begin{bmatrix} 3 + (-5) \\ 4 + 2 \end{bmatrix} = -\begin{bmatrix} -2 \\ 6 \end{bmatrix} = \begin{bmatrix} 2 \\ -6 \end{bmatrix}. \end{aligned}$$

The third rope pulls with $(2, -6)$ N, which has magnitude $\sqrt{4 + 36} = \sqrt{40} \approx 6.32$ N, pointing right and steeply down. Drawn tip to tail, the three forces form a closed triangle: you end where you began.

26.2.7. Alternative approach: magnitude-and-angle form

Mechanics problems often give vectors as "a speed and a bearing" rather than as components. You can add those directly using the law of cosines on the triangle, but that's error-prone and doesn't scale beyond two vectors. The reliable method is: **convert everything to components, add, convert back**. A vector of length r at angle θ above the positive x-axis has components

$$\mathbf{v} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix},$$

straight from the unit circle (Volume 1, Chapter 13). After adding, recover the length with Pythagoras and the angle with $\tan^{-1}$ (or better, `atan2`, which gets the quadrant right). **Choose components** whenever more than two vectors are involved, or whenever you'll compute.

Computer-science connection: game physics

Every physics engine, from a 2D platformer to a flight simulator, stores position, velocity and acceleration as vectors and updates them every frame with vector addition and scaling:

```python
import numpy as np

position = np.array([0.0, 0.0])        # metres
velocity = np.array([3.0, 4.0])        # m/s: thrown up and to the right
acceleration = np.array([0.0, -9.8])   # gravity, m/s^2
dt = 0.1                               # seconds per frame

for frame in range(3):
    # One frame of the simplest integrator (explicit Euler).
    velocity = velocity + acceleration * dt
    position = position + velocity * dt
    print(frame, position.round(3), velocity.round(3))
```

```
0 [0.3   0.302] [3.   3.02]
1 [0.6   0.506] [3.   2.04]
2 [0.9   0.612] [3.   1.06]
```

Gravity is the vector $(0, -9.8)$ m/s²; a jump adds an upward velocity vector; wind is a constant vector added every frame. The arithmetic is exactly this section's: component-wise addition and scalar multiplication. Real engines use more careful integrators (see Chapter 32), but the data structure is always a vector.



Drag the arrows

Two draggable arrows $\mathbf{a}$ and $\mathbf{b}$ start at the origin. The app draws $\mathbf{a} + \mathbf{b}$ both tip-to-tail and as the parallelogram diagonal, and shows the component arithmetic updating live. A slider for c shows $c\mathbf{a}$ stretching, shrinking, and flipping as c passes through zero.

Try it: Drag the arrows

26.3. Exercises: adding and scaling

1. **(Recognition)** Which of these are vectors and which are scalars: temperature, wind velocity, mass, the force of gravity on a book, the price of a book?

2. **(Application)** With $\mathbf{a} = (2, -1, 4)$ and $\mathbf{b} = (0, 3, -2)$, compute $3\mathbf{a} - 2\mathbf{b}$.

3. **(Application)** Points $P = (1, 5)$ and $Q = (4, 1)$. Find the vector $\overrightarrow{PQ}$ and the midpoint of PQ using vectors.

4. **(Problem solving)** In the aircraft example, what heading should the pilot point (air velocity of magnitude 200 km/h) so that the ground track is due north? What's the resulting ground speed?

▼ **Answers**

1. Vectors: wind velocity, the force of gravity (it points down). Scalars: temperature, mass, price.

2. $3\mathbf{a} = (6, -3, 12)$, $2\mathbf{b} = (0, 6, -4)$. Subtract component by component: $(6 - 0,\ -3 - 6,\ 12 - (-4)) = (6, -9, 16)$.

3. $\overrightarrow{PQ} = Q - P = (4 - 1, 1 - 5) = (3, -4)$. The midpoint is halfway along: $P + \tfrac{1}{2}\overrightarrow{PQ} = (1, 5) + (1.5, -2) = (2.5, 3)$. (Equivalently $\tfrac{1}{2}(P + Q)$.)

4. Let the air velocity be $\mathbf{p} = (p_x, p_y)$ with $p_x^2 + p_y^2 = 200^2$. Ground velocity $\mathbf{p} + (50, 0)$ must have zero x-component, so $p_x = -50$. Then $p_y = \sqrt{40\,000 - 2500} = \sqrt{37\,500} \approx 193.6$. The pilot points $\sin^{-1}(50/200) \approx 14.5°$ west of north, and the ground speed is about 193.6 km/h, slower than the airspeed because part of the engine's effort cancels the wind.

26.4. Length, distance, and unit vectors

26.4.1. The problem

A game character moves with velocity $(1, 0)$ when you press the right arrow and $(0, 1)$ when you press up. Press both, and the code adds them: $(1, 1)$. Players soon notice that moving diagonally is faster than moving straight. How much faster, and how do you fix it?

To answer, we need the length of a vector, and a way to keep a direction while setting the length to whatever we want.

26.4.2. Length from Pythagoras, in any dimension

In two dimensions, the arrow (v_1, v_2) is the hypotenuse of a right triangle with legs $|v_1|$ and $|v_2|$, so its length is $\sqrt{v_1^2 + v_2^2}$. For $(1, 1)$ that's $\sqrt{2} \approx 1.414$: diagonal movement is about 41% faster. That's the bug.

In three dimensions, apply Pythagoras twice. The vector (v_1, v_2, v_3) has a "shadow" $(v_1, v_2, 0)$ on the floor, of length $\sqrt{v_1^2 + v_2^2}$. The full vector rises vertically v_3 above the end of that shadow, making a second right triangle:

$$\begin{aligned} \|\mathbf{v}\|^2 &= \left(\sqrt{v_1^2 + v_2^2}\right)^2 + v_3^2 \qquad \text{(Pythagoras on the vertical triangle)} \\ &= v_1^2 + v_2^2 + v_3^2. \end{aligned}$$

Nothing stops the pattern there. Each new axis is perpendicular to all the previous ones, so each adds one more squared term. In n dimensions, **define** the length the same way:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} = \sqrt{\sum_{i=1}^{n} v_i^2}.$$



Magnitude (norm) and summation notation

The **magnitude**, **length** or **norm** of $\mathbf{v}$ is written $\|\mathbf{v}\|$ (double bars; some books use single bars $|\mathbf{v}|$). It's the **Euclidean norm**, also called the **L2 norm** and written $\|\mathbf{v}\|_2$ in machine-learning papers.

The symbol $\sum_{i=1}^{n} v_i^2$ is **summation notation**: Σ (capital Greek sigma, for "sum") means "add up", the **index** i runs from the value below (1) to the value above (n), and v_i^2 is the term being added for each i. So $\sum_{i=1}^{3} v_i^2 = v_1^2 + v_2^2 + v_3^2$. In code it's `sum(x*x for x in v)`.

The distance between two **points** P and Q is the length of the vector between them, $\|Q - P\|$. In two dimensions this is exactly the distance formula from Volume 1, Chapter 10 ; now it works in any dimension.

26.4.3. Unit vectors: direction without size

A **unit vector** has length 1. Dividing any nonzero vector by its own length produces a unit vector pointing the same way:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}.$$

This is called **normalising** $\mathbf{v}$. The hat notation $\hat{\mathbf{v}}$ ("v-hat") is the usual marker for a unit vector. Normalising throws away the magnitude and keeps the pure direction; you can then multiply by any length you like. The special unit vectors along the axes, $\mathbf{i} = (1,0,0), \mathbf{j} = (0,1,0), \mathbf{k} = (0,0,1)$, are the **standard basis vectors**, also written $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$.

Example 28. Procedure — normalising a vector

Input: a vector $\mathbf{v}$.

Steps:

1. Compute $\|\mathbf{v}\| = \sqrt{\sum v_i^2}$.
2. If $\|\mathbf{v}\| = 0$ (or is tiny compared with the precision of your data), stop: the zero vector has no direction. Report an error or return a sensible default.
3. Divide every component by $\|\mathbf{v}\|$.

Output: the unit vector $\hat{\mathbf{v}}$. Check: its length should be 1 (up to rounding).

26.4.4. Worked example

Normalise $\mathbf{v} = (2, -1, 2)$. *Try it:* Normalise (2, −1, 2) .

$$\begin{aligned} \|\mathbf{v}\| &= \sqrt{2^2 + (-1)^2 + 2^2} = \sqrt{4 + 1 + 4} = \sqrt{9} = 3, \\ \hat{\mathbf{v}} &= \tfrac{1}{3}(2, -1, 2) = \left(\tfrac{2}{3}, -\tfrac{1}{3}, \tfrac{2}{3}\right), \\ \textbf{check: } \|\hat{\mathbf{v}}\|^2 &= \tfrac{4}{9} + \tfrac{1}{9} + \tfrac{4}{9} = 1. \quad \checkmark \end{aligned}$$

And the game fix: before applying movement, normalise the input direction (when it isn't zero) and multiply by the desired speed s. Diagonal input $(1, 1)$ becomes $s \cdot (0.707, 0.707)$, whose length is s, the same as straight movement.

26.4.5. Where you'd use this

- **Everyday: your phone's step counter.** The accelerometer reports a 3D vector in m/s², but you might hold the phone in any orientation, so the individual components are meaningless on their own. The step counter uses the **length**. Standing still, a reading like $(3, 6, 7)$ has length $\sqrt{9 + 36 + 49} = \sqrt{94} \approx 9.7$, which is just gravity. Mid-stride, $(6, 12, 4)$ has length $\sqrt{36 + 144 + 16} = \sqrt{196} = 14$. Count the peaks of the length and you've counted steps, whichever pocket the phone is in.
- **Science and engineering:** a drone's speed from its velocity components, the resultant of several forces, and the distance between two atoms in a molecule (from their 3D coordinates) are all $\|\mathbf{v}\|$ or $\|Q - P\|$. Surveyors and GPS receivers compute 3D distances to satellites exactly this way.
- **Computing:** colour difference. Two pixels $(255, 100, 50)$ and $(250, 110, 40)$ in RGB differ by $(5, -10, 10)$, a distance of $\sqrt{25 + 100 + 100} = 15$; image editors use this (or a perceptual variant) for "magic wand" selection. The same distance, in more dimensions, drives **k-nearest-neighbour** classifiers and recommendation lookups. And unit vectors are everywhere in graphics: surface normals, camera directions and light directions are all normalised before use, as the Lambert shading sidebar below shows.



The zero vector cannot be normalised

$\mathbf{0}/\|\mathbf{0}\|$ is $0/0$. In code this produces `nan` (NumPy) or a `ZeroDivisionError` (plain Python), and a single `nan` silently poisons everything it touches: a character with `nan` position vanishes, a model with a `nan` weight outputs `nan` forever. Always guard normalisation, typically with `if norm > 1e-12`. A near-zero vector is almost as bad: its "direction" is mostly rounding noise.

26.5. Linear combinations and span

26.5.1. The problem

You have two moves available: $\mathbf{a} = (2, 1)$ and $\mathbf{b} = (1, -1)$, and you may use each any real number of times (including fractions and negatives). Can you reach the point $(7, 1)$? Can you reach **every** point?

26.5.2. Intuition

An expression like $x\mathbf{a} + y\mathbf{b}$, scaled vectors added together, is called a **linear combination** of $\mathbf{a}$ and $\mathbf{b}$. Think of $\mathbf{a}$ and $\mathbf{b}$ as two directions on a strange compass, and x, y as how far you travel along each. The **span** of $\mathbf{a}$ and $\mathbf{b}$ is the set of everywhere you can reach.

- One nonzero vector spans a **line** through the origin (all its multiples, as in the scalar-multiple diagram).
- Two vectors that point in genuinely different directions span a **plane**. In $\mathbb{R}^2$, that plane is everything.
- Two vectors that lie on the same line (one is a multiple of the other) span only that line. The second adds nothing new.
- Three vectors in $\mathbb{R}^3$ span all of space, unless one of them lies in the plane of the other two.

26.5.3. Worked example: reaching $(7, 1)$

We need $x(2, 1) + y(1, -1) = (7, 1)$. Comparing components gives a system of equations (Volume 1, Chapter 6): *Try it:* Reach (7, 1) .

$$\begin{aligned} 2x + y &= 7 \\ x - y &= 1. \end{aligned}$$

Adding the equations eliminates y: $3x = 8$, so $x = 8/3$. Then $y = x - 1 = 5/3$. Check: $\frac{8}{3}(2, 1) + \frac{5}{3}(1, -1) = (\frac{16}{3} + \frac{5}{3},\ \frac{8}{3} - \frac{5}{3}) = (7, 1)$. ✓

The same elimination works for **any** target (p, q): $x = (p + q)/3$, $y = (p - 2q)/3$. So $\mathbf{a}$ and $\mathbf{b}$ span the whole plane.

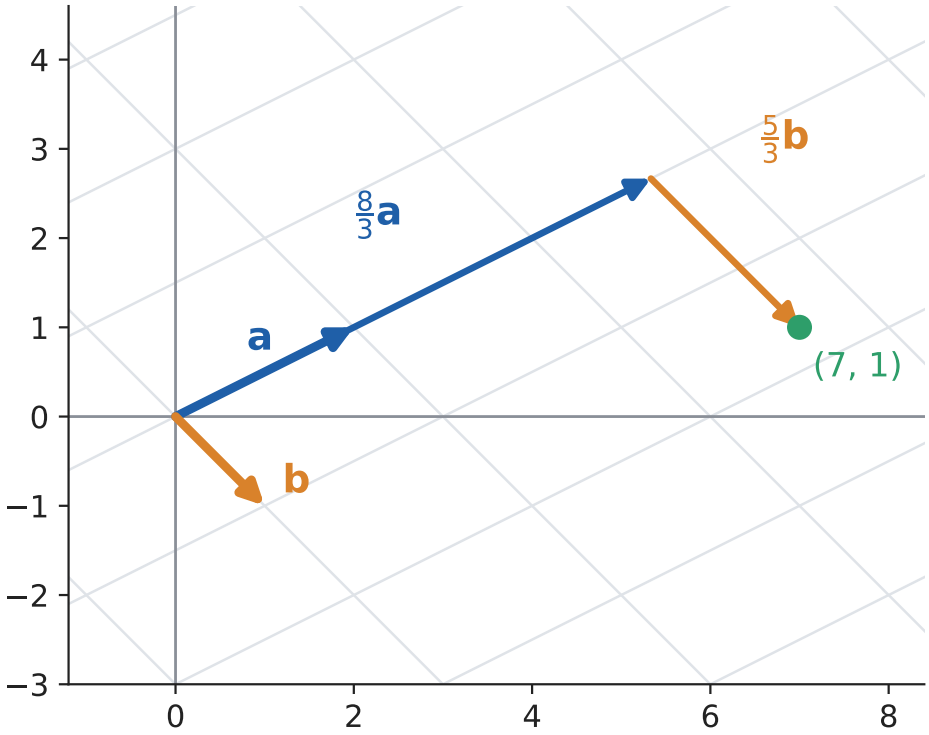


Figure 61. Reaching $(7, 1)$ with $\frac{8}{3}$ of $\mathbf{a}$ followed by $\frac{5}{3}$ of $\mathbf{b}$. The faint lines are the skewed grid of whole-number combinations of $\mathbf{a}$ and $\mathbf{b}$; because the grid covers the plane, every point can be reached.

Notice the deep fact hiding here: **asking whether a vector is a linear combination of others is the same as solving a system of linear equations.** That equivalence is the beating heart of linear algebra, and Chapter 27 builds on it.

Linear combination, span, independence, basis

- A **linear combination** of $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is any vector $c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_k\mathbf{v}_k$ with scalar **coefficients** c_i.
- The **span** of a set of vectors is the set of all their linear combinations.
- Vectors are **linearly independent** if none of them is a linear combination of the others: each contributes a genuinely new direction. Otherwise they're **linearly dependent**.
- A **basis** of a space is a linearly independent set that spans it. Every vector in the space is then a combination of basis vectors in exactly one way. The number of basis vectors is the **dimension** of the space.
- The **standard basis** of $\mathbb{R}^n$ is $\mathbf{e}_1, \ldots, \mathbf{e}_n$, and the components of a vector are precisely its coefficients in that basis: $(3, 4) = 3\mathbf{e}_1 + 4\mathbf{e}_2$.



Two vectors in 3D are not "enough"

Counting vectors is not the same as counting directions. Two vectors in $\mathbb{R}^3$ span at most a plane, never all of space. And $(1, 2, 3)$, $(2, 4, 6)$ and $(0, 1, 0)$ are three vectors but span only a plane, because the second is twice the first. When a feature vector in machine learning has 100 entries but they're all computed from 5 underlying measurements, the data really lives in (at most) a 5-dimensional subspace. Spotting that hidden low dimension is exactly what PCA does in Chapter 28.

26.5.4. Where you'd use this

- **Everyday: hitting a nutrition target.** Per 100 g, cooked lentils give about 9 g protein and 20 g carbohydrate, and cooked rice about 3 g protein and 28 g carbohydrate. Each food is a vector, (protein, carbs) = $(9, 20)$ and $(3, 28)$. To hit 27 g protein and 124 g carbohydrate you need $x(9, 20) + y(3, 28) = (27, 124)$, that is $9x + 3y = 27$ and $20x + 28y = 124$. The first gives $y = 9 - 3x$; substituting, $20x + 252 - 84x = 124$, so $x = 2$ and $y = 3$: 200 g of lentils and 300 g of rice. Diet-planning apps and animal-feed formulators solve exactly this, with dozens of nutrients.
- **Science: balancing a chemical equation.** Record each molecule as a vector of atom counts (C, H, O): $\mathrm{CH_4} = (1, 4, 0)$, $\mathrm{O_2} = (0, 0, 2)$, $\mathrm{CO_2} = (1, 0, 2)$, $\mathrm{H_2O} = (0, 2, 1)$. Balancing means finding coefficients with $1(1, 4, 0) + 2(0, 0, 2) = (1, 4, 4) = 1(1, 0, 2) + 2(0, 2, 1)$, which is $\mathrm{CH_4} + 2\mathrm{O_2} \to \mathrm{CO_2} + 2\mathrm{H_2O}$: a linear combination problem.
- **Computing:** every pixel colour is a linear combination $r\,\mathbf{red} + g\,\mathbf{green} + b\,\mathbf{blue}$ of three basis colours. JPEG goes further: each 8×8 block of an image is rewritten as a linear combination of 64 fixed "cosine pattern" basis images, and compression keeps only the coefficients that matter. Animation tweens and gradients use the combination $(1 - t)\mathbf{a} + t\mathbf{b}$ ("lerp") to slide smoothly from $\mathbf{a}$ at $t = 0$ to $\mathbf{b}$ at $t = 1$.



Combination sliders

Fix two vectors $\mathbf{a}$ and $\mathbf{b}$. Two sliders control x and y, and the app draws $x\mathbf{a} + y\mathbf{b}$ with a faint trail. Sweep the sliders and watch the trail fill the plane. Then drag $\mathbf{b}$ onto the line of $\mathbf{a}$ and watch the reachable set collapse to a line. A "target" mode places a random point and asks the learner to find the coefficients.

Try it: Combination sliders, collapsing span

26.6. The dot product

26.6.1. The problem

Two very different questions turn out to have the same answer.

A Mechanics question. You drag a sled 5 m across flat snow, pulling on a rope with a 10 N force angled 60° above the horizontal. How much work do you do? Work is "force times distance moved **in the direction of the force**", so only part of your pull counts.

A data question. Two documents, or two users, or two songs, are represented by vectors of numbers. How **similar** are they? We'd like a single number that's large when the vectors point the same way and small or negative when they don't.

Both questions ask: **how much do two vectors agree in direction, weighted by their sizes?** The tool that answers it is the dot product.

26.6.2. The algebraic definition

Multiply matching components and add up:

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = \sum_{i=1}^{n} a_i b_i.$$

The result is a **scalar**, not a vector, which is why it's also called the **scalar product** (or **inner product**). For example,

$$(3, 4) \cdot (4, -3) = 3 \cdot 4 + 4 \cdot (-3) = 12 - 12 = 0, \qquad (3, 4) \cdot (3, 4) = 9 + 16 = 25.$$

The second calculation shows an important link: **a vector dotted with itself is its length squared**, $\mathbf{v} \cdot \mathbf{v} = \|\mathbf{v}\|^2$. Directly from the definition, the dot product also satisfies

- $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$ (symmetric),
- $\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$ (distributes over addition),
- $(s\mathbf{a}) \cdot \mathbf{b} = s(\mathbf{a} \cdot \mathbf{b})$ (scalars pull out).

Each holds because it holds for every term $a_i b_i$ separately.

That's the mechanical definition. On its own it looks arbitrary: why multiply matching components? The answer is that this mechanical recipe secretly measures an **angle**.

26.6.3. Why it works: the geometric meaning



Law of cosines

In any triangle with sides a, b, c, where θ is the angle between sides a and b (opposite side c):

$$c^2 = a^2 + b^2 - 2ab\cos\theta.$$

When $\theta = 90°$, $\cos\theta = 0$ and it reduces to Pythagoras. The $-2ab\cos\theta$ term is the correction for the triangle not being right-angled. (Derived in Volume 1, Chapter 13 .)

Put the tails of $\mathbf{a}$ and $\mathbf{b}$ together, with angle θ between them. The third side of the triangle runs from the tip of $\mathbf{b}$ to the tip of $\mathbf{a}$, which is the vector $\mathbf{a} - \mathbf{b}$.

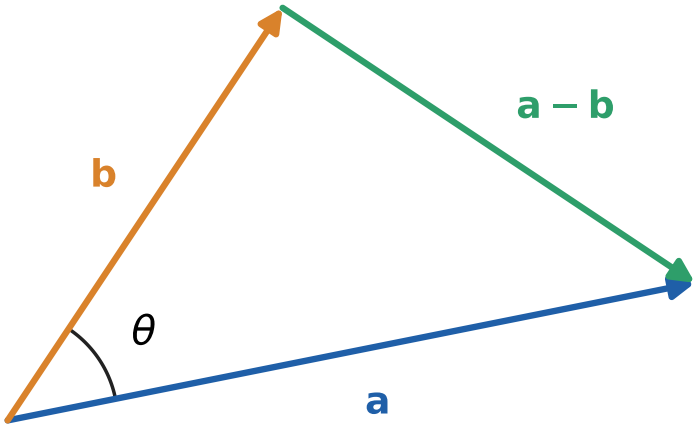


Figure 62. The triangle formed by $\mathbf{a}$, $\mathbf{b}$ *and* $\mathbf{a}-\mathbf{b}$.

Now compute the squared length of the third side in two ways.

Using the dot product algebra:

$$\begin{aligned}\|\mathbf{a}-\mathbf{b}\|^2 &= (\mathbf{a}-\mathbf{b})\cdot(\mathbf{a}-\mathbf{b}) && \text{(length squared is self-dot)}\\ &= \mathbf{a}\cdot\mathbf{a}-\mathbf{a}\cdot\mathbf{b}-\mathbf{b}\cdot\mathbf{a}+\mathbf{b}\cdot\mathbf{b} && \text{(distribute, like expanding brackets)}\\ &= \|\mathbf{a}\|^2+\|\mathbf{b}\|^2-2\,\mathbf{a}\cdot\mathbf{b} && \text{(symmetry: } \mathbf{a}\cdot\mathbf{b}=\mathbf{b}\cdot\mathbf{a}\text{).}\end{aligned}$$

Using the law of cosines with side lengths $\|\mathbf{a}\|$, $\|\mathbf{b}\|$, $\|\mathbf{a}-\mathbf{b}\|$:

$$\|\mathbf{a}-\mathbf{b}\|^2 = \|\mathbf{a}\|^2+\|\mathbf{b}\|^2-2\,\|\mathbf{a}\|\,\|\mathbf{b}\|\cos\theta.$$

The left sides are equal, so the right sides are equal. Cancel $\|\mathbf{a}\|^2+\|\mathbf{b}\|^2$ from both and divide by -2:

$$\boxed{\mathbf{a}\cdot\mathbf{b} = \|\mathbf{a}\|\,\|\mathbf{b}\|\cos\theta}$$

This is the central fact of the chapter. The "multiply and add" recipe, which a computer does in nanoseconds, equals "length times length times the cosine of the angle between", which is geometry. Two representations, one number.



Dot product

The **dot product** (scalar product, inner product) of two vectors of the same dimension is

$$\mathbf{a}\cdot\mathbf{b} = \sum_{i=1}^{n} a_i b_i = \|\mathbf{a}\|\,\|\mathbf{b}\|\cos\theta.$$

It is a scalar. Other notations: $\langle\mathbf{a},\mathbf{b}\rangle$ (mathematics, physics), $\mathbf{a}^{\mathsf{T}}\mathbf{b}$ (linear algebra, "a-transpose b"; see Chapter 27), and `a @ b` or `np.dot(a, b)` in NumPy.

26.6.4. What the sign tells you

Since lengths are positive, the sign of $\mathbf{a} \cdot \mathbf{b}$ is the sign of $\cos\theta$:

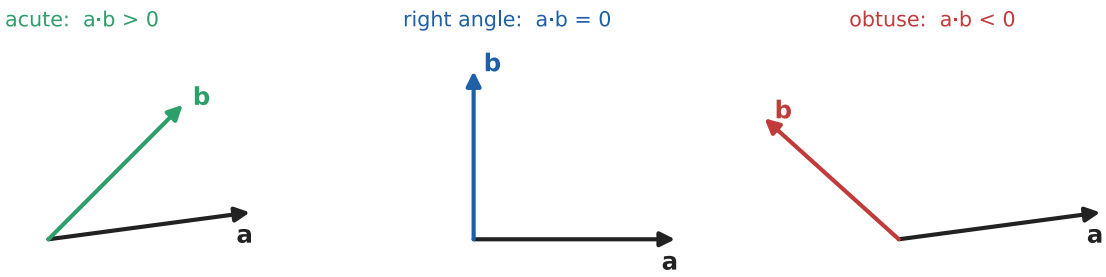


Figure 63. The sign of the dot product tracks the angle: positive when the vectors broadly agree, zero at a right angle, negative when they broadly oppose.

The middle case gets its own word. Vectors with $\mathbf{a} \cdot \mathbf{b} = 0$ are **orthogonal** (perpendicular, at right angles). Earlier we found $(3, 4) \cdot (4, -3) = 0$: those two arrows are perpendicular, which you can confirm by sketching them. This is an astonishingly cheap perpendicularity test: no angles, no trigonometry, just multiply and add. In 768 dimensions, where you can't sketch anything, it's the **only** practical test.



Orthogonal

Two vectors are **orthogonal** if their dot product is zero. For nonzero vectors this means they meet at 90°. (The zero vector is orthogonal to everything, by this definition.) "Orthogonal" is the general-dimension word; "perpendicular" is usual in 2D and 3D geometry; "normal" is used for a vector perpendicular to a surface. A set of vectors that are mutually orthogonal **and** all of length 1 is **orthonormal**; the standard basis is the classic example.

Example 29. Procedure — the angle between two vectors

Input: nonzero vectors $\mathbf{a}$, $\mathbf{b}$ of the same dimension.

Steps:

1. Compute $\mathbf{a} \cdot \mathbf{b} = \sum a_i b_i$.
2. Compute $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$.
3. Compute $\cos\theta = \dfrac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|}$. (In code, clamp the result into $[-1, 1]$ before the next step; rounding can push it to 1.0000000002.)
4. $\theta = \cos^{-1}(\cos\theta)$, which lies between $0°$ and $180°$.

Output: the angle θ. If only "same way / perpendicular / opposite" matters, stop after step 1 and read the sign.

26.6.5. Worked example: angle between two 3D vectors

Find the angle between $\mathbf{a} = (1, 2, 2)$ and $\mathbf{b} = (2, -1, 2)$. *Try it:* Angle between two 3D vectors .

$$\begin{aligned}
\mathbf{a} \cdot \mathbf{b} &= 1 \cdot 2 + 2 \cdot (-1) + 2 \cdot 2 = 2 - 2 + 4 = 4, \\
\|\mathbf{a}\| &= \sqrt{1 + 4 + 4} = 3, \qquad \|\mathbf{b}\| = \sqrt{4 + 1 + 4} = 3, \\
\cos\theta &= \frac{4}{3 \cdot 3} = \frac{4}{9} \approx 0.4444, \\
\theta &= \cos^{-1}(0.4444) \approx 63.6^\circ.
\end{aligned}$$

26.6.6. Worked example: work done dragging a sled

Work done by a constant force $\mathbf{F}$ over a displacement $\mathbf{d}$ is $W = \mathbf{F} \cdot \mathbf{d}$. With the rope at 60°:

$$\begin{aligned}
\mathbf{F} &= (10\cos 60^\circ,\ 10\sin 60^\circ) \approx (5,\ 8.66)\ \text{N}, \qquad \mathbf{d} = (5, 0)\ \text{m}, \\
W &= \mathbf{F} \cdot \mathbf{d} = 5 \cdot 5 + 8.66 \cdot 0 = 25\ \text{J}.
\end{aligned}$$

Check with the geometric form: $\|\mathbf{F}\|\,\|\mathbf{d}\|\cos 60^\circ = 10 \cdot 5 \cdot 0.5 = 25$ J. The vertical part of your pull (8.66 N) does no work because the sled doesn't move vertically: its dot product with the displacement is zero. That's the physical meaning of orthogonality, and it's why carrying a heavy bag across a level room does, by the physicist's definition, no work on the bag.

26.6.7. Where you'd use this

- **Everyday: a shopping bill is a dot product.** Quantities $\mathbf{q} = (3, 2, 1)$ (pints of milk, loaves, jars of coffee) and prices $\mathbf{p} = (1.20, 1.50, 4.00)$ give the total $\mathbf{q} \cdot \mathbf{p} = 3.60 + 3.00 + 4.00 = 10.60$. Any weighted total is a dot product: a course grade from weighted assignments, a nutrition label's calories from grams of fat, carbohydrate and protein, a portfolio's value from holdings and share prices.
- **Science and engineering:** work $\mathbf{F} \cdot \mathbf{d}$ (above) and power $\mathbf{F} \cdot \mathbf{v}$. A solar panel's output follows Lambert's cosine law: a 1.6 m² panel in 1000 W/m² sunshine collects 1600 W of sunlight when it faces the sun squarely, but only $1600\cos 60^\circ = 800$ W when the sun is 60° away from its normal. That's why solar farms use trackers that tilt the panels to follow the sun.
- **Computing:** a single artificial neuron computes $\mathbf{w} \cdot \mathbf{x} + b$ (Chapter 27), lighting uses $\mathbf{n} \cdot \mathbf{l}$ (below), and testing $\mathbf{a} \cdot \mathbf{b} = 0$ is the cheapest possible check for "perpendicular", used in collision detection and geometry code everywhere.



acos and floating point

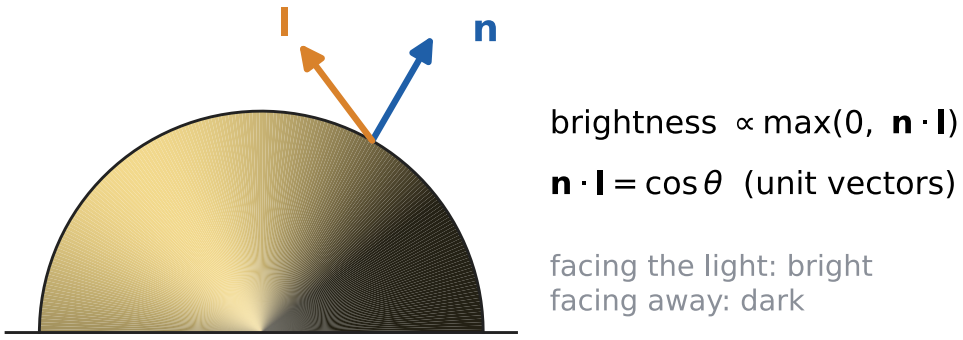
`math.acos(x)` raises `ValueError: math domain error` if `x` is even slightly outside $[-1, 1]$, and with nearly parallel vectors rounding regularly produces values like `1.0000000000000002`. Clamp first: `math.acos(max(-1.0, min(1.0, c)))`. Separately, for very small angles $\cos^{-1}$ is numerically poor, since $\cos\theta$ barely changes near 0. If you need tiny angles accurately, use `atan2(norm(cross(a, b)), dot(a, b))` in 3D. More on this in Chapter 32.

Computer-science connection: Lambert shading

Why does the side of a ball facing a lamp look bright and the far side dark? A patch of matte surface receives light in proportion to how squarely it faces the light source. If $\mathbf{n}$ is the unit **surface normal** (pointing straight out of the surface) and $\mathbf{l}$ is the unit vector towards the light, the brightness is

$$I = I_{\text{light}} \cdot k_d \cdot \max(0,\ \mathbf{n} \cdot \mathbf{l}),$$

where k_d is the surface's colour (its **diffuse reflectance**). Face the light head-on: $\mathbf{n} \cdot \mathbf{l} = 1$, full brightness. Tilt by 60°: $\cos 60° = 0.5$, half brightness. Face away: the dot product goes negative and the $\max$ clamps it to zero, shadow. This is **Lambert's cosine law** (1760), and one line of it, `max(0.0, dot(n, l))`, runs billions of times per second in every GPU's fragment shaders.



26.7. Projections

26.7.1. The problem

A 4 kg box rests on a ramp inclined at 30°. Gravity pulls straight down. How much of that pull acts **along** the ramp (trying to slide the box) and how much acts **into** the ramp (pressing it against the surface)?

Equivalently: given a vector $\mathbf{a}$ and a direction $\mathbf{b}$, split $\mathbf{a}$ into a part along $\mathbf{b}$ and a part perpendicular to $\mathbf{b}$. Mechanics calls this **resolving** a force. Linear algebra calls the first part the **projection** of $\mathbf{a}$ onto $\mathbf{b}$.

26.7.2. Intuition: the shadow

Shine a light perpendicular to the line of $\mathbf{b}$. The shadow that $\mathbf{a}$ casts on that line is its projection. What's left over, from the tip of the shadow up to the tip of $\mathbf{a}$, is perpendicular to $\mathbf{b}$.

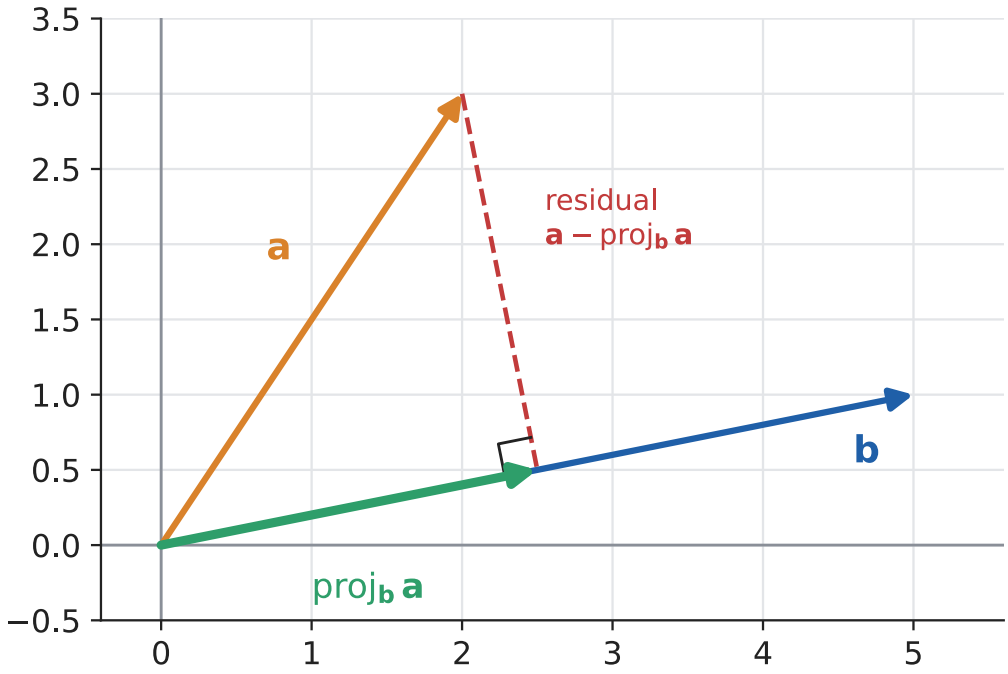


Figure 64. Projecting $\mathbf{a} = (2, 3)$ onto $\mathbf{b} = (5, 1)$. The projection (green) lies along $\mathbf{b}$; the residual (red) meets $\mathbf{b}$ at a right angle.

26.7.3. Why it works: deriving the formula

Let $\hat{\mathbf{b}} = \mathbf{b}/\|\mathbf{b}\|$ be the unit vector along $\mathbf{b}$. In the right triangle formed by $\mathbf{a}$, its shadow and the residual, $\mathbf{a}$ is the hypotenuse and the shadow is the side adjacent to angle θ. So the shadow's (signed) length is

$$\begin{aligned} \|\mathbf{a}\| \cos\theta &= \frac{\|\mathbf{a}\|\,\|\mathbf{b}\| \cos\theta}{\|\mathbf{b}\|} && \text{(multiply and divide by } \|\mathbf{b}\|) \\ &= \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|} = \mathbf{a} \cdot \hat{\mathbf{b}}. && \text{(geometric dot product)} \end{aligned}$$

That number is the **scalar projection** (or **component** of $\mathbf{a}$ along $\mathbf{b}$). It's negative if $\mathbf{a}$ points partly backwards. To get the projection as a vector, multiply this length by the unit direction:

$$\operatorname{proj}_{\mathbf{b}} \mathbf{a} = (\mathbf{a} \cdot \hat{\mathbf{b}})\,\hat{\mathbf{b}} = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{b} \cdot \mathbf{b}}\,\mathbf{b}.$$

The second form avoids square roots, which is handy by hand and in code. The perpendicular part is whatever's left: $\mathbf{a} - \operatorname{proj}_{\mathbf{b}} \mathbf{a}$.

Projection

- **Scalar projection** of $\mathbf{a}$ onto $\mathbf{b}$: $\mathbf{a} \cdot \hat{\mathbf{b}} = \dfrac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|}$, a signed length.
- **Vector projection**: $\operatorname{proj}_{\mathbf{b}} \mathbf{a} = \dfrac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{b} \cdot \mathbf{b}} \mathbf{b}$, a vector along $\mathbf{b}$.
- **Orthogonal decomposition**: $\mathbf{a} = \operatorname{proj}_{\mathbf{b}} \mathbf{a} + (\mathbf{a} - \operatorname{proj}_{\mathbf{b}} \mathbf{a})$, one part parallel to $\mathbf{b}$, one perpendicular.

A key reading: **dotting with a unit vector measures the component in that direction.** That's why $\mathbf{v} \cdot \mathbf{i}$ is just v_1.

Example 30. Procedure — resolve a vector along a direction

Input: a vector $\mathbf{a}$ and a nonzero direction vector $\mathbf{b}$.

Steps:

1. Compute $\mathbf{a} \cdot \mathbf{b}$ and $\mathbf{b} \cdot \mathbf{b}$.
2. Compute the coefficient $k = (\mathbf{a} \cdot \mathbf{b})/(\mathbf{b} \cdot \mathbf{b})$.
3. Parallel part: $\mathbf{p} = k\,\mathbf{b}$.
4. Perpendicular part: $\mathbf{r} = \mathbf{a} - \mathbf{p}$.
5. Check: $\mathbf{r} \cdot \mathbf{b}$ should be 0 (up to rounding).

Output: $\mathbf{p}$ (along $\mathbf{b}$) and $\mathbf{r}$ (perpendicular), with $\mathbf{a} = \mathbf{p} + \mathbf{r}$.

26.7.4. Worked example: the diagram's numbers

Project $\mathbf{a} = (2, 3)$ onto $\mathbf{b} = (5, 1)$. *Try it:* Project a onto b .

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= 2 \cdot 5 + 3 \cdot 1 = 13, \qquad \mathbf{b} \cdot \mathbf{b} = 25 + 1 = 26, \\ k &= \tfrac{13}{26} = \tfrac{1}{2}, \\ \mathbf{p} &= \tfrac{1}{2}(5, 1) = (2.5,\ 0.5), \\ \mathbf{r} &= (2, 3) - (2.5,\ 0.5) = (-0.5,\ 2.5), \\ \text{check: } \mathbf{r} \cdot \mathbf{b} &= -0.5 \cdot 5 + 2.5 \cdot 1 = -2.5 + 2.5 = 0. \quad \checkmark \end{aligned}$$

In NumPy, where `@` between two 1-D arrays is the dot product:

```python
import numpy as np

a = np.array([2.0, 3.0])
b = np.array([5.0, 1.0])

def project(a, b):
    """Vector projection of a onto b."""
    return (a @ b) / (b @ b) * b

p = project(a, b)
r = a - p
print(p)            # the shadow of a on b's line
print(r)            # what's left over
print(r @ b)        # residual is perpendicular to b
```

```
[2.5 0.5]
[-0.5  2.5]
0.0
```

26.7.5. Worked example: the box on the ramp

Take x horizontal and y vertical. The weight is $\mathbf{W} = (0, -4 \times 9.8) = (0, -39.2)$ N. A unit vector pointing **up** the 30° ramp is $\hat{\mathbf{u}} = (\cos 30°, \sin 30°) \approx (0.866, 0.5)$, and a unit vector perpendicular to the ramp, pointing out of its surface, is $\hat{\mathbf{n}} = (-\sin 30°, \cos 30°) \approx (-0.5, 0.866)$. (Check: $\hat{\mathbf{u}} \cdot \hat{\mathbf{n}} = -0.433 + 0.433 = 0$.)

$$\begin{aligned} \mathbf{W} \cdot \hat{\mathbf{u}} &= 0 \cdot 0.866 + (-39.2)(0.5) = -19.6 \text{ N} && \text{(19.6 N down the slope)} \\ \mathbf{W} \cdot \hat{\mathbf{n}} &= 0 \cdot (-0.5) + (-39.2)(0.866) \approx -33.9 \text{ N} && \text{(33.9 N into the slope)} \end{aligned}$$

These are the familiar Mechanics results $mg \sin\theta$ and $mg \cos\theta$, now obtained without drawing a single similar triangle. The ramp pushes back with a normal reaction of 33.9 N; friction must supply up to 19.6 N to stop the box sliding.

26.7.6. Choosing a method

For resolving forces in a 2D Mechanics problem, the trigonometric shortcuts ($mg \sin\theta$, $mg \cos\theta$) are fastest **if** you draw the diagram correctly. The dot-product method is mechanical and immune to "was it sine or cosine?" errors, works in 3D and beyond, and is what you'd write in code. Use trig for quick sketches; use projection for anything you'll compute or anything not in a plane.

26.7.7. Where you'd use this

- **Everyday and aviation: headwind and crosswind.** A pilot landing on a runway, or a cyclist on a straight road, cares about two pieces of the wind: the part along the direction of travel and the part across it. A 30-knot wind blowing at 40° to the runway has a headwind component $30 \cos 40° \approx 23.0$ knots and a crosswind component $30 \sin 40° \approx 19.3$ knots. That second number decides whether a small aircraft may land (many have a crosswind limit around 15–20 knots).

- **Science:** resolving forces on slopes, cables and bridges (as above); in surveying, splitting a measured displacement into its horizontal and vertical parts.
- **Computing:** in a game, when a character runs into a wall you remove the into-the-wall part of its velocity and keep the projection along the wall, so it **slides** instead of stopping dead. Fitting a straight line to data by least squares (Volume 3, Chapter 31) is a projection too: the fitted values are the shadow of the data vector on the space of possible straight-line predictions, and the residuals are perpendicular to it.

Computer-science connection: bouncing off walls

When a ball with velocity $\mathbf{v}$ hits a wall with unit normal $\hat{\mathbf{n}}$, the part of $\mathbf{v}$ along the normal reverses and the part along the wall is unchanged. The along-normal part is $(\mathbf{v}\cdot\hat{\mathbf{n}})\hat{\mathbf{n}}$; subtracting it once removes it and subtracting it twice reverses it:

$$\mathbf{v}_{\text{new}} = \mathbf{v} - 2(\mathbf{v}\cdot\hat{\mathbf{n}})\,\hat{\mathbf{n}}.$$

For a ball moving left, $\mathbf{v} = (-2, 0)$, hitting a 45° wall with $\hat{\mathbf{n}} = (1, 1)/\sqrt{2}$: $\mathbf{v}\cdot\hat{\mathbf{n}} = -2/\sqrt{2} = -\sqrt{2}$, so $\mathbf{v}_{\text{new}} = (-2, 0) + 2\sqrt{2}\cdot(1, 1)/\sqrt{2} = (-2, 0) + (2, 2) = (0, 2)$. It goes straight up, as a mirror at 45° should. The same formula reflects light rays in ray tracers (GLSL even has a built-in `reflect(v, n)`).

Projection also powers **Gram–Schmidt**: to build an orthonormal basis from arbitrary vectors, take each vector, subtract its projections onto all the previous ones (leaving only the genuinely new direction), then normalise. That's how the QR decomposition, used inside least-squares solvers, is built.



Resolve a force

Drag a vector $\mathbf{a}$ and a direction $\mathbf{b}$. The app shows the shadow of $\mathbf{a}$ on the line of $\mathbf{b}$, the perpendicular residual with its right-angle marker, and the numbers $\mathbf{a}\cdot\mathbf{b}$, k and θ. Rotate $\mathbf{a}$ past 90° and watch the projection flip to point backwards as the dot product goes negative. A "ramp" mode tilts a slope and resolves the box's weight live.

Try it: Resolve a force

26.8. Exercises: magnitude, dot product, projection

1. **(Recognition)** Without computing an angle, say whether the angle between $(3, -1)$ and $(1, 2)$ is acute, right, or obtuse.
2. **(Application)** Find a unit vector in the direction of $(4, -4, 2)$.
3. **(Application)** Find the angle between $(3, -1)$ and $(1, 2)$ to the nearest degree.
4. **(Application)** Decompose $\mathbf{a} = (4, 2)$ into parts parallel and perpendicular to $\mathbf{b} = (1, 1)$.

5. **(Problem solving)** Find all vectors (x, y) orthogonal to $(2, 5)$, and give the one of length 1 with positive x-component.
6. **(Problem solving)** Show that for any vectors, $\|\mathbf{a}+\mathbf{b}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2$ exactly when $\mathbf{a}\cdot\mathbf{b} = 0$. What famous theorem is this?

▼ **Answers**

1. $(3)(1) + (-1)(2) = 1 > 0$, so the angle is acute (but only just).
2. $\|(4,-4,2)\| = \sqrt{16+16+4} = \sqrt{36} = 6$. Unit vector: $(4,-4,2)/6 = (\frac{2}{3}, -\frac{2}{3}, \frac{1}{3})$.
3. Dot product 1; lengths $\sqrt{10}$ and $\sqrt{5}$; $\cos\theta = 1/\sqrt{50} \approx 0.1414$; $\theta \approx 81.9° \approx 82°$.
4. $\mathbf{a}\cdot\mathbf{b} = 6$, $\mathbf{b}\cdot\mathbf{b} = 2$, $k = 3$. Parallel: $(3, 3)$. Perpendicular: $(4, 2) - (3, 3) = (1, -1)$. Check: $(1,-1)\cdot(1,1) = 0$. ✓
5. $2x + 5y = 0$, so $y = -\frac{2}{5}x$: all multiples of $(5, -2)$ (a line through the origin). Length $\sqrt{29}$, so the unit vector is $(5,-2)/\sqrt{29} \approx (0.928, -0.371)$. (Handy trick: in 2D, (a, b) is always perpendicular to $(b, -a)$ and $(-b, a)$.)
6. Expand: $\|\mathbf{a}+\mathbf{b}\|^2 = (\mathbf{a}+\mathbf{b})\cdot(\mathbf{a}+\mathbf{b}) = \|\mathbf{a}\|^2 + 2\,\mathbf{a}\cdot\mathbf{b} + \|\mathbf{b}\|^2$. This equals $\|\mathbf{a}\|^2 + \|\mathbf{b}\|^2$ exactly when $2\,\mathbf{a}\cdot\mathbf{b} = 0$. It's Pythagoras's theorem (and its converse), valid in any number of dimensions.

26.9. Cosine similarity, embeddings, and search

26.9.1. The problem

A search engine holds a million documents. A user types a query. Which documents are most relevant?

This is the founding problem of **information retrieval**, and since the 1960s the dominant answer has been geometric: turn every document and every query into a vector, and rank documents by how closely their vectors **point the same way** as the query's.

26.9.2. Documents as vectors

The simplest representation is a **bag of words**. Fix a vocabulary of n words. A document becomes a vector in $\mathbb{R}^n$ whose i-th component counts how often word i appears. With only two vocabulary words, "matrix" and "graph", we can draw it:

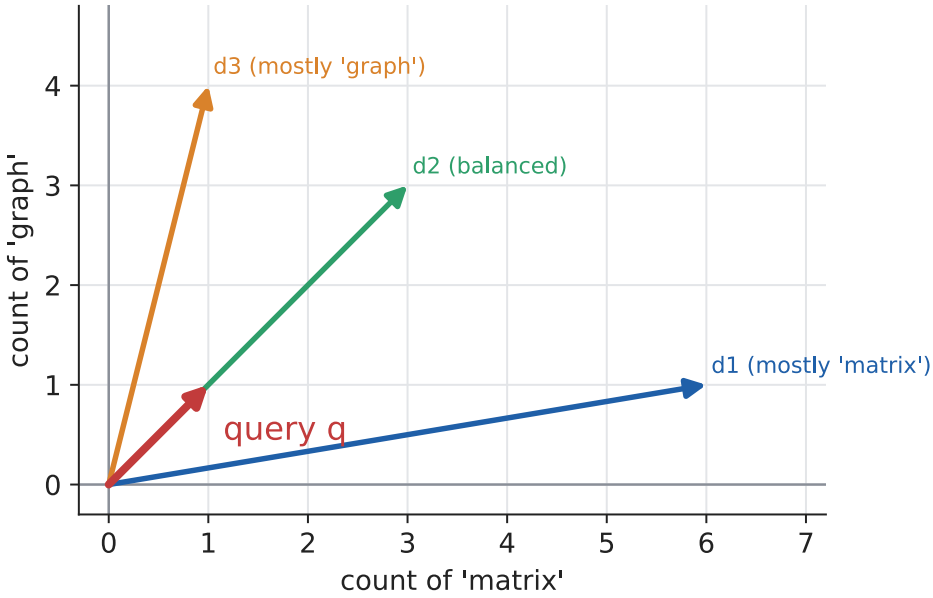


Figure 65. Documents and a query as vectors in a two-word space. Document $\mathbf{d_2} = (3, 3)$ points in exactly the same direction as the query $\mathbf{q} = (1, 1)$, so its cosine similarity is 1; $\mathbf{d_1} = (6, 1)$ is longer and has the larger raw dot product (7 against 6), but points well away from the query.

Should we rank by the raw dot product $\mathbf{q} \cdot \mathbf{d}$? It rewards documents that share words with the query, which is good. But $\mathbf{q} \cdot \mathbf{d} = \|\mathbf{q}\| \, \|\mathbf{d}\| \cos\theta$ also grows with $\|\mathbf{d}\|$, so a long document beats a short one just by being long. Paste a document into itself ten times and its score goes up tenfold, without becoming any more relevant.

The fix is to divide out the lengths, leaving only the angle. That's **cosine similarity**:

$$\cos_\mathrm{sim}(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\| \, \|\mathbf{d}\|} = \cos\theta.$$

It lies between -1 and 1 (between 0 and 1 for count vectors, which have no negative entries), and depends only on direction.

26.9.3. Worked example: length bias

Query $\mathbf{q} = (1, 1)$ (the words "matrix graph"). Three documents: $A = (10, 10)$, a long document using both words equally; $B = (2, 0)$, a short one about matrices only; $C = (1, 1)$, a short one using both. *Try it:* Dot product versus cosine .

$$\mathbf{q} \cdot A = 20, \quad \cos(\mathbf{q}, A) = \frac{20}{\sqrt{2} \cdot \sqrt{200}} = \frac{20}{20} = 1,$$

$$\mathbf{q} \cdot B = 2, \quad \cos(\mathbf{q}, B) = \frac{2}{\sqrt{2} \cdot 2} \approx 0.707,$$

$$\mathbf{q} \cdot C = 2, \quad \cos(\mathbf{q}, C) = \frac{2}{\sqrt{2} \cdot \sqrt{2}} = 1.$$

The raw dot product says A is ten times as relevant as C; cosine similarity correctly says they're equally on-topic, and both beat B, which covers only half the query.

26.9.4. TF-IDF: weighting the components

Raw counts have another flaw: "the" appears everywhere and dominates every vector while telling you nothing. **TF-IDF** reweights each component by how **informative** the word is. A word's **document frequency** $\mathrm{df}(w)$ is the number of documents containing it; out of N documents its **inverse document frequency** is $\mathrm{idf}(w) = \log(N/\mathrm{df}(w))$, which is zero for a word in every document and large for a rare one. Each component becomes term frequency times IDF. Everything else, including cosine ranking, stays the same. *Try it:* Rank with TF-IDF .

PYTHON

```
import math
from collections import Counter

corpus = [
    "the matrix rotates the vector",
    "the graph has a vertex and an edge",
    "a vector is an arrow and a vector is a list",
]
tokenised = [doc.split() for doc in corpus]
vocab = sorted({w for doc in tokenised for w in doc})
N = len(corpus)
df = Counter(w for doc in tokenised for w in set(doc))      # document frequency
idf = {w: math.log(N / df[w]) for w in vocab}

def tfidf(tokens):
    tf = Counter(tokens)
    return [tf[w] * idf[w] for w in vocab]

vectors = [tfidf(doc) for doc in tokenised]
query = tfidf("vector arrow".split())

def cosine(a, b):
    num = sum(x * y for x, y in zip(a, b))
    den = math.sqrt(sum(x * x for x in a)) * math.sqrt(sum(y * y for y in b))
    return num / den if den else 0.0

for doc, vec in zip(corpus, vectors):
    print(f"{cosine(query, vec):.3f}  {doc}")
```

```
0.078  the matrix rotates the vector
0.000  the graph has a vertex and an edge
0.421  a vector is an arrow and a vector is a list
```

The third document wins: it contains both query words, and "arrow" is rare (high IDF). The first scores a little for "vector". The second shares no query words and scores zero, its vector orthogonal to the query's.



TF-IDF and cosine similarity

- **Bag of words**: a document as a vector of word counts, ignoring order.
- **TF-IDF** (term frequency–inverse document frequency): each count multiplied by $\log(N/\text{df})$, so common words count for little and rare words for a lot.
- **Cosine similarity**: $\dfrac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\,\|\mathbf{b}\|}$, the cosine of the angle between two vectors; it ignores length. **Cosine distance** is $1 -$ **cosine similarity**.

26.9.5. From $(3, 4)$ to 768 dimensions: embeddings

Bag-of-words vectors have a blind spot. "Car" and "automobile" are different vocabulary entries, so the corresponding axes are orthogonal, and a query for one scores zero against a document using only the other. The vectors capture **spelling**, not **meaning**.

Modern systems use **embeddings**: vectors produced by a trained neural network, designed so that texts with similar meanings get vectors pointing in similar directions. A typical sentence embedding has 384, 768, 1024 or more components. Here is the important point, and it's worth slowing down for: **mathematically, an embedding is exactly the same kind of object as $(3, 4)$.** It's a list of real numbers. Its length is $\sqrt{\sum v_i^2}$, with 768 terms instead of 2. Its dot product with another embedding is $\sum a_i b_i$, with 768 products instead of 2. The angle between two embeddings is $\cos^{-1}$ of the normalised dot product, defined by the very formula we derived from the law of cosines. Nothing in our derivation used the fact that we were in two dimensions; we only used the algebraic rules of the dot product. You can't **draw** a 768-dimensional arrow, but every calculation you'd do on the drawing still works.

What's different is what the components **mean**. In $(3, 4)$ the components are "east" and "north". In an embedding, no single component has a clean meaning; meaning lives in **directions**, combinations of many components. Similar meanings point similar ways; that's all the geometry needs.

26.9.6. Reading $\mathbf{q}\cdot\mathbf{d}$

You'll meet the expression $\mathbf{q}\cdot\mathbf{d}$ constantly in information-retrieval and machine-learning papers. You can now unpack it completely:

- $\mathbf{q}$ is the query's vector and $\mathbf{d}$ a document's vector, in the same space (same dimension, same meaning of each axis).
- $\mathbf{q}\cdot\mathbf{d} = \sum_i q_i d_i$ is computed by multiplying matching components and adding: n multiplications and $n - 1$ additions.
- Geometrically it's $\|\mathbf{q}\|\,\|\mathbf{d}\|\cos\theta$: large when the vectors are long and aligned.
- Embedding systems usually **normalise** every vector to length 1 when they're stored. Then $\mathbf{q}\cdot\mathbf{d} = \cos\theta$ exactly: the dot product **is** cosine similarity, without the divisions. That's why papers often write the dot product and mean the cosine.
- For unit vectors, Euclidean distance carries the same information too:

$$\|\mathbf{q}-\mathbf{d}\|^2 = \|\mathbf{q}\|^2 + \|\mathbf{d}\|^2 - 2\,\mathbf{q}\cdot\mathbf{d} = 2 - 2\cos\theta,$$

so ranking by largest cosine, largest dot product, or smallest distance gives the **same** order. Which "metric" a vector database asks you to pick matters only for unnormalised vectors.

26.9.7. Worked example: semantic search in NumPy

Here are five toy documents with 4-dimensional "embeddings" (real ones come from a model, but the search step is identical). Storing the documents as the rows of a 2-D array lets NumPy compute all five dot products in one operation.

PYTHON

```python
import numpy as np

# Five toy "documents", each already embedded as a 4-dimensional vector.
docs = ["cat care guide", "kitten adoption", "stock market news",
        "interest rates rise", "dog training tips"]
D = np.array([
    [0.9, 0.1, 0.0, 0.3],
    [0.8, 0.2, 0.1, 0.1],
    [0.0, 0.9, 0.4, 0.0],
    [0.1, 0.8, 0.6, 0.0],
    [0.6, 0.0, 0.1, 0.8],
])
q = np.array([0.85, 0.05, 0.05, 0.2])         # "how do I look after a cat?"

# Normalise every row and the query to length 1.
Dn = D / np.linalg.norm(D, axis=1, keepdims=True)
qn = q / np.linalg.norm(q)

scores = Dn @ qn                    # five dot products in one operation
for i in np.argsort(-scores):       # highest score first
    print(f"{scores[i]:.3f}  {docs[i]}")
```

```
0.993  cat care guide
0.975  kitten adoption
0.767  dog training tips
0.176  interest rates rise
0.075  stock market news
```

Note the ranking: the pet documents cluster near the query; the finance ones point elsewhere. `Dn @ qn` is a matrix–vector product, which is just "dot the query with every row". We'll see in Chapter 27 that this is one of the two ways to read matrix–vector multiplication.

Example 31. Procedure — vector (semantic) search by cosine similarity

Input: document vectors $\mathbf{d}_1, \ldots, \mathbf{d}_N$, a query vector $\mathbf{q}$ (same dimension), and k, the number of results wanted.

Steps:

1. **Index time:** normalise every document vector to unit length and store them as rows of a matrix D.
2. **Query time:** normalise $\mathbf{q}$.
3. Compute the score vector $\mathbf{s} = D\hat{\mathbf{q}}$, i.e. $s_j = \hat{\mathbf{d}}_j \cdot \hat{\mathbf{q}}$ for every document j.
4. Select the k largest scores (a partial sort or heap, $O(N \log k)$).

Output: the top-k documents with their cosine similarities.

26.9.8. Choosing a similarity measure

- **Cosine similarity** (or dot product of normalised vectors): the default for text, both TF-IDF and embeddings. Use it when only direction (topic, meaning) matters.
- **Raw dot product**: use it when magnitude carries real signal, for instance in recommendation systems where a longer user vector means a more confident preference, or when the model was trained with dot-product scores (some retrieval models are).
- **Euclidean distance**: natural when the vectors are positions in a space with real units (pixels, metres), as in nearest-neighbour classification of physical measurements. For unit vectors it ranks identically to cosine.

The rule of thumb: **use the similarity your embedding model was trained with**. Its documentation will say.

26.9.9. Where you'd use this (beyond search)

- **Everyday: "people like you also liked".** Three friends rate four films out of 5: Ana $(5, 4, 1, 0)$, Ben $(4, 5, 0, 1)$, Cal $(1, 0, 5, 4)$. All three vectors have length $\sqrt{42}$, so $\cos(\text{Ana}, \text{Ben}) = 40/42 \approx 0.95$ and $\cos(\text{Ana}, \text{Cal}) = 10/42 \approx 0.24$. Ben is Ana's taste twin, so a film Ben loved that Ana hasn't seen is a good recommendation. Early recommender systems (Amazon's item-to-item recommendations, the Netflix Prize entries) used exactly this.
- **Science:** chemists compare molecules by cosine-like similarity of their "fingerprint" vectors when screening for drug candidates, and biologists compare gene-expression profiles across thousands of genes to find genes that switch on together.
- **Computing:** plagiarism and near-duplicate detection, clustering news stories, face recognition (two photos match when their face embeddings have cosine above a threshold), and retrieval-augmented generation for chatbots, which fetches the passages whose embeddings point closest to the question's.

Computer-science connection: vector databases and NumPy vectorisation

Brute-force search computes N dot products of dimension n: $O(Nn)$ work per query. For ten million 768-dimensional vectors that's about 7.7 billion multiply-adds, under a second on a GPU but too slow for every keystroke. **Vector databases** (FAISS, pgvector, Milvus, Elasticsearch's dense-vector fields, and many more) use **approximate nearest-neighbour** (ANN) indexes such as HNSW graphs (see Volume 3, Chapter 25) or product quantisation to find the top results while examining only a small fraction of the vectors, trading a little recall for orders of magnitude in speed. The underlying task is called **maximum inner product search** (MIPS).

At a smaller scale, the same lesson applies to Python. A loop of 768 multiplications in pure Python costs tens of microseconds; `a @ b` in NumPy hands the whole job to compiled code that uses the CPU's SIMD instructions to multiply several numbers per clock cycle. "Vectorising" code, replacing loops over components with whole-array operations, is the practical payoff of thinking in vectors rather than in lists of separate numbers.



King - man + woman ≈ queen

In 2013 Tomas Mikolov and colleagues at Google published **word2vec**, which learned 300-dimensional word vectors from billions of words of news text. They found that vector **arithmetic** captured analogies: the vector $\mathbf{king} - \mathbf{man} + \mathbf{woman}$ was closer (by cosine similarity) to **queen** than to any other word, and $\mathbf{Paris} - \mathbf{France} + \mathbf{Italy}$ landed near **Rome**. The **direction** from "man" to "woman" roughly encoded gender; the direction from a country to its capital encoded "capital of". Tip-to-tail addition, the rule we drew for walking across a field, turned out to model relationships between meanings. (Later work showed the effect is noisier than the famous examples suggest, and that these directions also encode social biases present in the training text, which is its own important lesson.)

26.10. High-dimensional intuition, and where it fails

26.10.1. The problem

Everything we've done extends algebraically to 768 dimensions. But our **intuition** was trained in two and three. Which mental pictures survive the trip, and which mislead?

26.10.2. Random directions are nearly perpendicular

Pick two random directions in the plane. The angle between them is equally likely to be anything, so their cosine similarity is spread widely between -1 and 1. Now pick two random directions in 768 dimensions. Their cosine similarity is almost always within a few hundredths of zero: **random high-dimensional vectors are nearly orthogonal**.

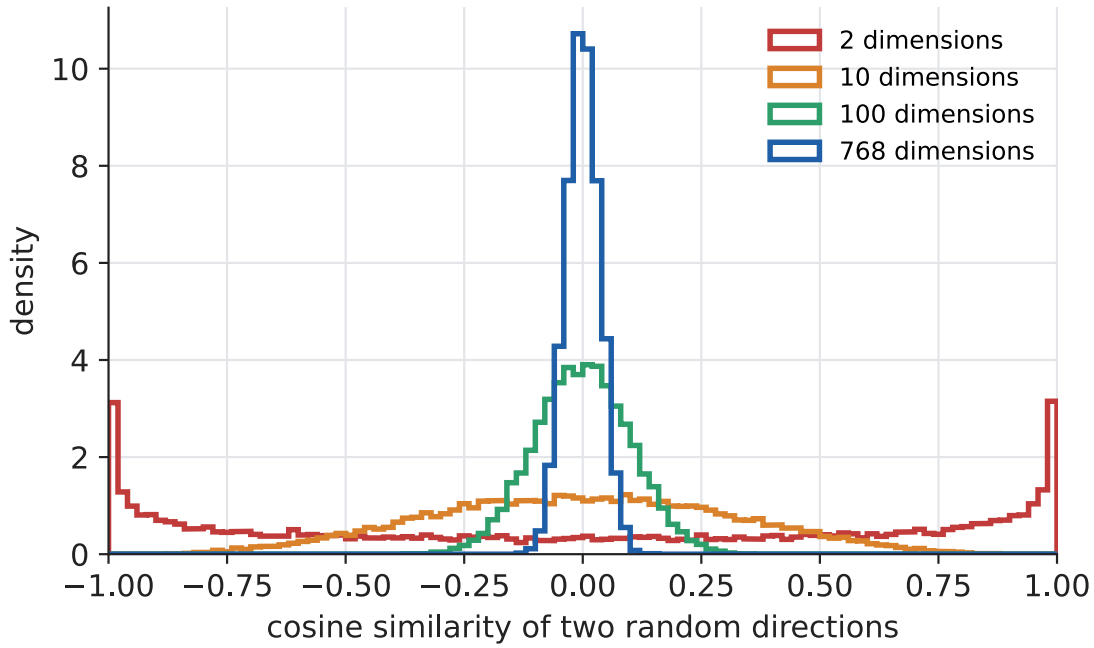


Figure 66. Cosine similarity between pairs of random directions. In 2D, any value is common; in 768 dimensions, random vectors are almost always within about ± 0.1 of perpendicular.

Why? The dot product of two random unit vectors is a sum of n small terms, each as likely to be positive as negative. Like a random walk, the positives and negatives mostly cancel. The typical size of such a sum, relative to the maximum possible, shrinks like $1/\sqrt{n}$ (a fact you'll be able to prove after Volume 3, Chapter 30). You can see it directly: *Try it:* Random cosines in 2D .

PYTHON

```python
import numpy as np

rng = np.random.default_rng(42)
for d in [2, 3, 10, 100, 768, 4096]:
    X = rng.standard_normal((5000, d))
    Y = rng.standard_normal((5000, d))
    cos = np.sum(X * Y, axis=1) / (np.linalg.norm(X, axis=1) * np.linalg.norm(Y, axis=1))
    print(f"d={d:5d}  std of cos = {cos.std():.3f}   1/sqrt(d) = {1/np.sqrt(d):.3f}")
```

```
d=    2  std of cos = 0.709   1/sqrt(d) = 0.707
d=    3  std of cos = 0.576   1/sqrt(d) = 0.577
d=   10  std of cos = 0.310   1/sqrt(d) = 0.316
d=  100  std of cos = 0.103   1/sqrt(d) = 0.100
d=  768  std of cos = 0.036   1/sqrt(d) = 0.036
d= 4096  std of cos = 0.016   1/sqrt(d) = 0.016
```

This is good news and bad news. Good: in high dimensions there's room for an enormous number of nearly independent directions, so a 768-dimensional space can encode many distinct concepts without them interfering. Bad: if your vectors are close to random, all similarities look alike, and "similar" becomes meaningless.



Three-dimensional intuition in high dimensions

- **A cosine of 0.3 can be significant.** In 768 dimensions, two unrelated vectors typically have $|\cos\theta| < 0.1$. Don't judge embedding similarities against a 2D mental scale where 0.3 means "pretty different". Equally, real embedding models are not random: their vectors often all sit in a narrow cone, so even unrelated texts may score 0.6 or more. Judge scores **relative to each other** for a given model, not against an absolute threshold.
- **Distances concentrate.** For random points in high dimensions, the nearest and farthest neighbours of a point are at almost the same distance. Nearest-neighbour search then becomes fragile. This is one face of the **curse of dimensionality**.
- **Volume lives near the surface.** In n dimensions, the fraction of a ball's volume within the outer 1% of its radius is $1 - 0.99^n$: about 2% in 2D but over 99.9% when $n = 768$. "Typical" high-dimensional points are near the edge, not the middle.
- **Pictures still help, cautiously.** Projecting to 2D (with PCA, Chapter 28) is invaluable for spotting structure, but it can make far-apart points look close. Treat 2D plots of embeddings as hints, not evidence.



Spin up the dimension

A slider sets the dimension n from 2 to 2000. The app draws 1000 random vector pairs and plots the histogram of their cosine similarities, with the curve narrowing as n grows. A second panel shows the distribution of distances from one random point to all others, so the learner can watch "nearest" and "farthest" converge. A third loads real sentence embeddings for the learner's own sentences and compares their similarity scores against the random baseline.

Try it: Spin up the dimension

26.11. The cross product (three dimensions only)

26.11.1. The problem

A 3D model is made of triangles. To light a triangle with Lambert's law, you need its normal vector: a vector perpendicular to the triangle's surface. You have two edge vectors $\mathbf{a}$ and $\mathbf{b}$ lying in the triangle. How do you find a vector perpendicular to both?

26.11.2. The formula and its meaning

In three dimensions (and only there) there's a product of two vectors that **is** a vector, perpendicular to both inputs:

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}.$$

A mnemonic for the pattern: for the first component, ignore the first entries and cross-multiply the second and third ($a_2b_3 - a_3b_2$); the other components cycle the indices $1 \to 2 \to 3 \to 1$. (In Chapter 27 you'll see it written as a determinant.)

Its geometric properties are what matter:

- **Direction:** perpendicular to both $\mathbf{a}$ and $\mathbf{b}$, following the **right-hand rule**: point your right hand's fingers along $\mathbf{a}$, curl them towards $\mathbf{b}$, and your thumb points along $\mathbf{a} \times \mathbf{b}$.
- **Magnitude:** $\|\mathbf{a} \times \mathbf{b}\| = \|\mathbf{a}\|\,\|\mathbf{b}\| \sin\theta$, which is the **area of the parallelogram** the two vectors span. Parallel vectors ($\theta = 0$) give the zero vector.
- **Order matters:** $\mathbf{b} \times \mathbf{a} = -(\mathbf{a} \times \mathbf{b})$. Swapping flips the direction (it's **anticommutative**).

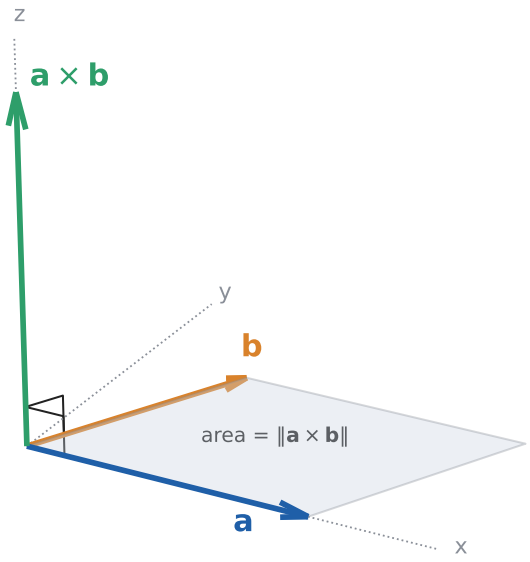


Figure 67. The cross product $\mathbf{a} \times \mathbf{b}$ is perpendicular to the parallelogram spanned by $\mathbf{a}$ and $\mathbf{b}$, and its length equals that parallelogram's area.



Cross product

The **cross product** (vector product) $\mathbf{a} \times \mathbf{b}$ of two 3D vectors is a 3D vector perpendicular to both, of length $\|\mathbf{a}\|\,\|\mathbf{b}\| \sin\theta$, oriented by the right-hand rule. Contrast the **dot** product, which works in any dimension and returns a **scalar** involving $\cos\theta$. In unit-vector form: $\mathbf{i} \times \mathbf{j} = \mathbf{k}, \mathbf{j} \times \mathbf{k} = \mathbf{i}, \mathbf{k} \times \mathbf{i} = \mathbf{j}$.

26.11.3. Worked example

Compute $(1, 2, 3) \times (4, 5, 6)$ and check the result. *Try it:* Cross product (1,2,3)×(4,5,6) .

$$\begin{aligned}
\text{1st: } & a_2b_3 - a_3b_2 = 2\cdot 6 - 3\cdot 5 = 12 - 15 = -3,\\
\text{2nd: } & a_3b_1 - a_1b_3 = 3\cdot 4 - 1\cdot 6 = 12 - 6 = 6,\\
\text{3rd: } & a_1b_2 - a_2b_1 = 1\cdot 5 - 2\cdot 4 = 5 - 8 = -3,\\
\mathbf{a}\times\mathbf{b} &= (-3,\ 6,\ -3),\\
\text{check: } & (-3,6,-3)\cdot(1,2,3) = -3+12-9 = 0, \quad (-3,6,-3)\cdot(4,5,6) = -12+30-18 = 0.\ \checkmark
\end{aligned}$$

The parallelogram has area $\sqrt{9+36+9} = \sqrt{54} \approx 7.35$, so the triangle with those two edges has half that, about 3.67. NumPy agrees: `np.cross([1, 2, 3], [4, 5, 6])` returns `array([-3, 6, -3])`.

Applications: torque, normals, culling

In Mechanics, the **moment** (torque) of a force $\mathbf{F}$ applied at position $\mathbf{r}$ from a pivot is $\boldsymbol{\tau} = \mathbf{r}\times\mathbf{F}$: its length is the familiar "force times perpendicular distance", and its direction is the axis of rotation. Pull with 40 N at right angles on the end of a 25 cm spanner: $(0.25,0,0)\times(0,40,0) = (0,0,10)$, a torque of 10 N·m about the bolt's axis. A wheel-nut specification of "110 N·m" tells you how hard to pull on a wrench of a given length. In electromagnetism, the force on a charge moving through a magnetic field is $q\,\mathbf{v}\times\mathbf{B}$, perpendicular to both motion and field; that sideways push is what turns an electric motor.

In graphics, the normal of a triangle with corners P_0, P_1, P_2 is $(P_1 - P_0)\times(P_2 - P_0)$, normalised. Because the cross product's direction depends on order, the **winding order** of the corners (clockwise or anticlockwise as seen by the camera) decides whether the normal faces the camera. GPUs exploit this for **back-face culling**: if $\mathbf{n}\cdot\mathbf{v} > 0$, where $\mathbf{v}$ points from the camera to the triangle, the triangle faces away and is skipped, roughly halving the work for closed objects. Get the winding wrong in a model file and half your mesh vanishes.



Only in 3D (and, oddly, 7D)

There is no cross product of two vectors in 2D, 4D, or 768D giving a vector perpendicular to both with these properties (a product of this kind exists only in 3 and 7 dimensions). In 2D, people often use the **scalar** $a_1b_2 - a_2b_1$ ("2D cross product" in game code), which is the z-component of the 3D cross product of $(a_1,a_2,0)$ and $(b_1,b_2,0)$; its sign says whether $\mathbf{b}$ is anticlockwise from $\mathbf{a}$. In higher dimensions, the job of "area and orientation" is done by the **determinant**, which we meet in Chapter 27.

26.12. Exercises: similarity and cross products

1. **(Recognition)** Two embeddings are normalised to length 1 and have dot product 0.82. What's their cosine similarity, and what's the Euclidean distance between them?
2. **(Application)** Compute the cosine similarity of the word-count vectors $(2,0,1)$ and $(1,1,1)$.
3. **(Application)** Find a normal vector to the triangle with corners $(0,0,0)$, $(1,0,0)$, $(0,2,2)$, and the triangle's area.

4. **(Problem solving)** A search engine stores unnormalised TF-IDF vectors and ranks by dot product. A user complains that long, rambling pages always win. Explain the cause geometrically and propose a fix.

▼ **Answers**

1. For unit vectors the dot product **is** the cosine similarity: 0.82. Distance: $\sqrt{2 - 2(0.82)} = \sqrt{0.36} = 0.6$.
2. Dot product $2 + 0 + 1 = 3$. Lengths $\sqrt{5}$ and $\sqrt{3}$. Cosine: $3/\sqrt{15} \approx 0.775$.
3. Edges $\mathbf{a} = (1, 0, 0)$, $\mathbf{b} = (0, 2, 2)$.
   $\mathbf{a} \times \mathbf{b} = (0 \cdot 2 - 0 \cdot 2,\ 0 \cdot 0 - 1 \cdot 2,\ 1 \cdot 2 - 0 \cdot 0) = (0, -2, 2)$. Check: dot with $\mathbf{a}$ is 0, with $\mathbf{b}$ is $-4 + 4 = 0$. Parallelogram area $\sqrt{8} = 2\sqrt{2}$, so the triangle's area is $\sqrt{2} \approx 1.41$.
4. $\mathbf{q} \cdot \mathbf{d} = \|\mathbf{q}\|\,\|\mathbf{d}\| \cos\theta$. Long documents have large $\|\mathbf{d}\|$, which inflates the score regardless of the angle (topic match). Fix: normalise document vectors to unit length at index time (rank by cosine similarity), or use a scoring function with explicit length normalisation such as BM25.

26.13. Summary

- A **vector** has magnitude and direction. Picture it as an arrow (geometry) or store it as a list of components (computation); the two views always agree.
- Vectors **add** component-wise (tip-to-tail), and **scalar multiplication** stretches, shrinks or flips them. Linear combinations of vectors **span** lines, planes, or whole spaces; checking whether a vector is in the span means solving a linear system.
- **Length** is Pythagoras in any dimension: $\|\mathbf{v}\| = \sqrt{\sum v_i^2}$. **Normalising** gives a unit vector; never normalise the zero vector.
- The **dot product** $\sum a_i b_i$ equals $\|\mathbf{a}\|\,\|\mathbf{b}\| \cos\theta$ (from the law of cosines). Zero means **orthogonal**. It computes angles, work, projections, Lambert shading and reflections.
- **Projection** onto $\mathbf{b}$: $\frac{\mathbf{a}\cdot\mathbf{b}}{\mathbf{b}\cdot\mathbf{b}}\mathbf{b}$, leaving a perpendicular residual.
- **Cosine similarity** compares directions while ignoring length; for unit vectors it equals the dot product, and Euclidean distance ranks identically. Search engines, TF-IDF and embedding models all rank by it.
- An embedding is the same mathematical object as $(3, 4)$, with more components. But high-dimensional space has surprises: random vectors are nearly orthogonal and distances concentrate.
- The **cross product** (3D only) gives a vector perpendicular to two others, with length equal to their parallelogram's area.

26.14. Interactive tools

Browser tools on the companion website that go with this chapter. Each example link in the chapter opens one of them with that example already set up.

- Vectors playground
- The maths of machine learning

26.15. Mastery check

1. Let $\mathbf{u} = (1, -2, 2)$ and $\mathbf{v} = (3, 0, 4)$. Compute $\mathbf{u} + 2\mathbf{v}$, $\|\mathbf{u}\|$, $\|\mathbf{v}\|$, $\mathbf{u} \cdot \mathbf{v}$, and the angle between $\mathbf{u}$ and $\mathbf{v}$.
2. Are $(2, 1)$ and $(-1, 2)$ orthogonal? Justify with one calculation.
3. Find the projection of $(4, 2)$ onto $(1, 1)$, and the unit vector in the direction of $(6, 8)$.
4. Write $(5, 4)$ as a linear combination of $(1, 1)$ and $(1, -1)$.
5. Derive $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$ from the law of cosines, in three or four lines.
6. A surface has unit normal $\mathbf{n} = (0, 0.6, 0.8)$ and the unit light direction is $\mathbf{l} = (0, 0, 1)$. What fraction of full Lambert brightness does it receive? What if $\mathbf{l} = (0, -0.8, -0.6)$?
7. In a 1000-dimensional embedding space, about how large would you expect the cosine similarity of two **unrelated random** vectors to be? Why does this matter when interpreting a similarity score of 0.25?
8. Compute $(2, 0, 1) \times (1, 3, 0)$ and verify it's perpendicular to both inputs.

Decision rule: All correct without hesitation → move on to Chapter 27. Stuck on 1–4 → reread Section 26.2, Section 26.4 and Section 26.5. Stuck on 5–6 → reread Section 26.6 and Section 26.7. Stuck on 7 → reread Section 26.9 and Section 26.10. Stuck on 8 → reread Section 26.11.

▼ **Mastery check answers**

1. $\mathbf{u} + 2\mathbf{v} = (1 + 6, -2 + 0, 2 + 8) = (7, -2, 10)$. $\|\mathbf{u}\| = \sqrt{1 + 4 + 4} = 3$; $\|\mathbf{v}\| = \sqrt{9 + 0 + 16} = 5$. $\mathbf{u} \cdot \mathbf{v} = 3 + 0 + 8 = 11$. $\cos\theta = 11/15 \approx 0.733$, so $\theta \approx 42.8°$.
2. $(2)(-1) + (1)(2) = 0$. Yes, orthogonal.
3. $\frac{(4,2)\cdot(1,1)}{(1,1)\cdot(1,1)}(1, 1) = \frac{6}{2}(1, 1) = (3, 3)$. $\|(6, 8)\| = 10$, so the unit vector is $(0.6, 0.8)$.
4. $x(1, 1) + y(1, -1) = (5, 4)$ gives $x + y = 5$, $x - y = 4$. Adding: $2x = 9$, $x = 4.5$; then $y = 0.5$. So $(5, 4) = 4.5(1, 1) + 0.5(1, -1)$.
5. $\|\mathbf{a} - \mathbf{b}\|^2 = (\mathbf{a} - \mathbf{b}) \cdot (\mathbf{a} - \mathbf{b}) = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - 2\,\mathbf{a} \cdot \mathbf{b}$. The law of cosines on the triangle with sides $\mathbf{a}$, $\mathbf{b}$, $\mathbf{a} - \mathbf{b}$ gives $\|\mathbf{a} - \mathbf{b}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - 2\|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$. Equate and cancel: $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$.
6. $\mathbf{n} \cdot \mathbf{l} = 0 + 0 + 0.8 = 0.8$: 80% brightness (the surface is tilted about $36.9°$ from the light). With the second light, $\mathbf{n} \cdot \mathbf{l} = 0 - 0.48 - 0.48 = -0.96 < 0$, so $\max(0, \cdot) = 0$: the light is behind the surface, and it's in shadow.

7. Typically about $1/\sqrt{1000} \approx 0.03$ in size, almost always below 0.1. So 0.25 is far above chance for **random** vectors and is not "nearly unrelated" as 2D intuition might suggest. But real models' vectors aren't random (they often share a common direction), so compare against other scores from the same model rather than reading 0.25 in isolation.

8. $(0 \cdot 0 - 1 \cdot 3,\ 1 \cdot 1 - 2 \cdot 0,\ 2 \cdot 3 - 0 \cdot 1) = (-3, 1, 6)$. Checks:
   $(-3, 1, 6) \cdot (2, 0, 1) = -6 + 0 + 6 = 0$; $(-3, 1, 6) \cdot (1, 3, 0) = -3 + 3 + 0 = 0$. ✓

Chapter 27. Matrices as Transformations

Most people first meet a matrix as a rectangular grid of numbers with a peculiar multiplication rule to memorise. That's backwards. A matrix is a machine that moves space around: it can rotate it, stretch it, shear it, reflect it, or flatten it, and the grid of numbers is just a compact record of what the machine does. Once you see that, the strange multiplication rule becomes obvious (it's "do one transformation, then another"), the inverse becomes "undo", the determinant becomes "how much does area change", and singular matrices become "transformations that squash space and can't be reversed". This chapter builds every one of those ideas from pictures first, then turns them into procedures and code, and finishes in the places matrices earn their living: graphics pipelines and neural networks.

Where this chapter sits

You need: vectors, linear combinations, span and the dot product (Chapter 26); systems of linear equations and elimination (Volume 1, Chapter 6); sine and cosine on the unit circle (Volume 1, Chapter 13); the idea of a function as an input-to-output rule (Volume 1, Chapter 11). No calculus.

This chapter unlocks: eigenvectors, diagonalisation, PageRank, PCA and the SVD (Chapter 28); linear regression and least squares (Volume 3, Chapter 31); the Jacobian and the multivariable chain rule (Chapter 20); conditioning and numerical stability (Chapter 32); and the linear-algebra core of machine learning (Volume 3, Chapter 33).

Why this chapter matters. Whenever many outputs depend linearly on many inputs, a matrix is the bookkeeping: ingredients needed for a bakery's orders, the joints of a robot arm, the pixels of a rotated photo, every vertex on a game screen, every layer of a neural network. Learn to read a matrix as a transformation and all of these become the same idea.

27.1. Transformations of space

27.1.1. The problem

A game needs to rotate a spaceship sprite by 30°. A photo editor needs to skew an image so a slanted building looks upright. A 3D engine must take every vertex of a model, place it in a world, look at it from a camera, and flatten the result onto a screen, 60 times a second, for millions of vertices. A neural network must turn a 768-number embedding into a 3072-number hidden representation.

Each of these is a **function that takes vectors in and produces vectors out**. We want a representation of such functions that is compact, fast to apply, easy to combine, and easy to reverse. For a very important class of them, the **linear** transformations, that representation is the matrix.

27.1.2. Intuition: watch the grid

The best way to understand a transformation of the plane is to apply it to **everything at once** and watch what happens to the coordinate grid. Here are six transformations applied to the grid, to an asymmetric letter F (so you can see flips and turns), and to the two standard basis vectors $\mathbf{e_1} = (1, 0)$ (green) and $\mathbf{e_2} = (0, 1)$ (red).

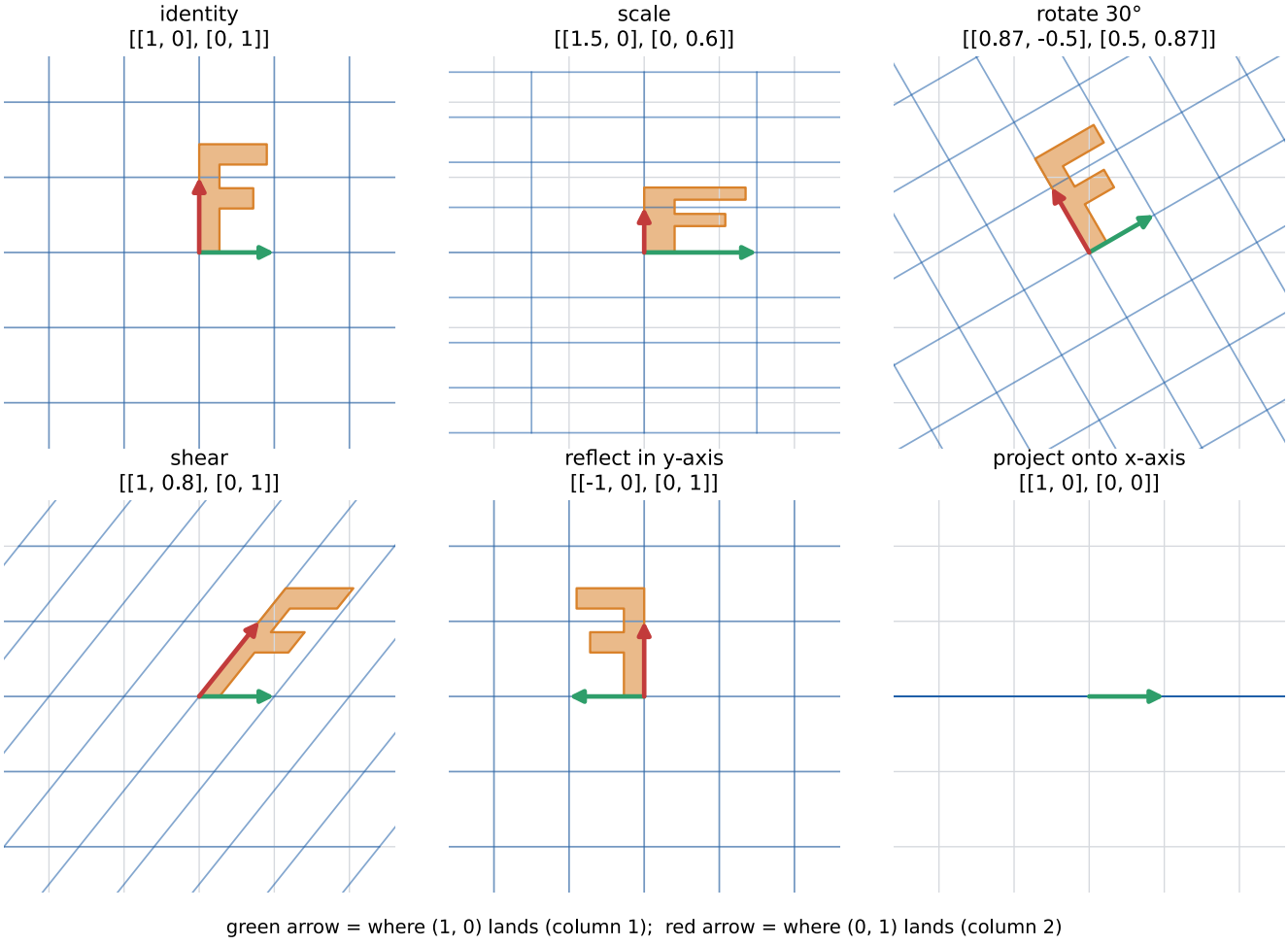


Figure 68. Six linear transformations of the plane. Grey: the original grid. Blue: the transformed grid. The green and red arrows show where $\mathbf{e_1}$ and $\mathbf{e_2}$ land; the numbers above each panel are the matrix, written row by row.

Look at what all six have in common:

- The origin stays put.
- Straight grid lines stay straight (none of them bend into curves).
- Parallel grid lines stay parallel, and evenly spaced lines stay evenly spaced.

These are the visual signatures of a **linear transformation**. A rotation keeps squares as squares; a scale stretches them into rectangles; a shear slants them into parallelograms; a reflection flips the F; a projection flattens the whole plane onto a line. But in every case, a regular grid becomes a regular grid (possibly a flattened one).

27.1.3. Why it works: the two rules of linearity

The pictures have a precise algebraic meaning. A function T from vectors to vectors is **linear** if, for all vectors $\mathbf{u}, \mathbf{v}$ and scalars c,

$$T(\mathbf{u}+\mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v}) \qquad \text{and} \qquad T(c\,\mathbf{v}) = c\,T(\mathbf{v}).$$

In words: **transforming a sum gives the sum of the transforms, and transforming a scaled vector gives the scaled transform**. Tip-to-tail parallelograms map to tip-to-tail parallelograms. Taking $c = 0$ shows $T(\mathbf{0}) = \mathbf{0}$: the origin is fixed. And "evenly spaced lines stay evenly spaced" is the second rule applied to the multiples $\mathbf{1v}, \mathbf{2v}, \mathbf{3v}, \ldots$.

Rotations, scalings, shears, reflections and projections through the origin are all linear. Some important transformations are **not**: translation (sliding everything by a fixed vector) moves the origin, so it fails the test. We'll rescue translation with a clever trick in Section 27.11.



Linear transformation

A **linear transformation** (or **linear map**) $T : \mathbb{R}^n \to \mathbb{R}^m$ is a function from n-dimensional vectors to m-dimensional vectors satisfying $T(\mathbf{u}+\mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$ and $T(c\mathbf{v}) = cT(\mathbf{v})$. The arrow notation $T : \mathbb{R}^n \to \mathbb{R}^m$ reads "T maps R-n to R-m": the **domain** is $\mathbb{R}^n$, the **codomain** $\mathbb{R}^m$. Geometrically: the origin is fixed, and grid lines stay straight, parallel and evenly spaced.



Change a matrix, watch the grid

Four sliders (or draggable number fields) set the entries of a 2×2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$. The app shows the coordinate grid, the unit square, an asymmetric F and the two basis arrows, and animates them smoothly from the identity to the chosen transformation. The learner can also drag the tips of the green and red arrows directly, with the matrix entries updating to match (the columns **are** the arrow tips). Readouts show the determinant (with the unit square's area highlighted) and flag when it reaches zero and the plane collapses to a line. Preset buttons: rotate, scale, shear, reflect, project. A "compose" mode applies two matrices in sequence and lets the learner swap their order.

Try it: Change a matrix, watch the grid

27.2. The columns tell you everything

27.2.1. The problem

A linear transformation acts on infinitely many vectors. How can a finite list of numbers possibly describe it?

27.2.2. Intuition: follow the basis vectors

Every vector in the plane is a combination of the basis vectors: $(x, y) = x\,\mathbf{e}_1 + y\,\mathbf{e}_2$. Linearity says a transformation respects combinations. So

$$\begin{aligned} T(x, y) &= T(x\,\mathbf{e}_1 + y\,\mathbf{e}_2) \\ &= T(x\,\mathbf{e}_1) + T(y\,\mathbf{e}_2) && \text{(rule 1: sums)} \\ &= x\,T(\mathbf{e}_1) + y\,T(\mathbf{e}_2). && \text{(rule 2: scalars)} \end{aligned}$$

That's the whole secret. **Once you know where $\mathbf{e}_1$ and $\mathbf{e}_2$ land, you know where everything lands.** A linear transformation of the plane is completely determined by two vectors, four numbers. Write those two landing spots side by side as columns, and you have the matrix:

$$A = \Big[\, T(\mathbf{e}_1) \;\Big|\; T(\mathbf{e}_2) \,\Big].$$

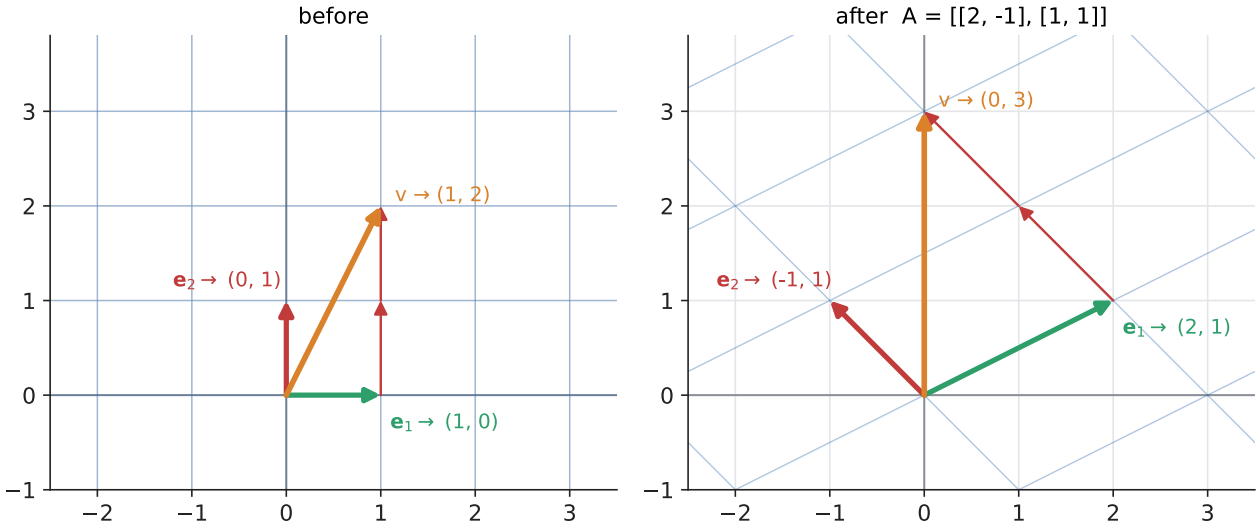


Figure 69. The matrix $A = \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}$ sends $\mathbf{e}_1$ to its first column $(2, 1)$ and $\mathbf{e}_2$ to its second column $(-1, 1)$. The vector $\mathbf{v} = (1, 2) = 1\mathbf{e}_1 + 2\mathbf{e}_2$ therefore lands at $1(2, 1) + 2(-1, 1) = (0, 3)$.

Matrix

A **matrix** is a rectangular array of numbers. An $m \times n$ matrix ("m by n") has m **rows** and n **columns**; rows first, always. The **entry** in row i, column j is written a_{ij} (so a_{12} is row 1, column 2). As a transformation, an $m \times n$ matrix takes n-dimensional vectors to m-dimensional vectors: column j is where the j-th basis vector lands. A **square** matrix has $m = n$ and maps a space to itself. The word is Latin for "womb" (the thing from which something grows), chosen by James Sylvester in 1850.

27.2.3. Matrix–vector multiplication: two views

Applying a matrix to a vector is written $A\mathbf{x}$, and from the derivation above we already know what it must be: **a linear combination of the columns, weighted by the entries of $\mathbf{x}$.**

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = x \begin{bmatrix} a \\ c \end{bmatrix} + y \begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}.$$

Now read the final result row by row. The top entry $ax + by$ is the dot product of the **first row** (a, b) with $\mathbf{x}$; the bottom entry is the dot product of the second row with $\mathbf{x}$. So there are two equally correct ways to compute $A\mathbf{x}$:

- **Column view:** $A\mathbf{x}$ is a combination of the columns of A. This is the geometric view: "where do the basis vectors go, and how much of each do I take?"
- **Row view:** entry i of $A\mathbf{x}$ is (row i of A) $\cdot$ $\mathbf{x}$. This is the computational view, and it's what you do by hand. It's also what the semantic search in Chapter 26 did: stacking document vectors as rows of D made $D\mathbf{q}$ the list of all document–query dot products.

Example 32. Procedure — multiply a matrix by a vector

Input: an $m \times n$ matrix A and a vector $\mathbf{x}$ with n entries.

Steps:

1. Check that the number of columns of A equals the length of $\mathbf{x}$. If not, the product is undefined.
2. For each row $i = 1, \ldots, m$: multiply each entry a_{ij} by x_j and add, giving $(A\mathbf{x})_i = \sum_{j=1}^{n} a_{ij} x_j$.
3. (Equivalently: add up $x_1 \times$ column 1, $x_2 \times$ column 2, ..., $x_n \times$ column n.)

Output: the vector $A\mathbf{x}$ with m entries. Cost: mn multiplications.

Try it: Step through matrix × vector .

27.2.4. Worked example

Compute $A\mathbf{v}$ for $A = \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}$ and $\mathbf{v} = (1, 2)$, both ways.

$$\text{Row view: } A\mathbf{v} = \begin{bmatrix} (2)(1) + (-1)(2) \\ (1)(1) + (1)(2) \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix},$$

$$\text{Column view: } A\mathbf{v} = 1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 - 2 \\ 1 + 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}.$$

Same answer, of course, and it matches the diagram. In code the two views are two loop orders over the same arithmetic:

```python
def matvec_rows(A, x):
    """Each output entry is (row of A) . x"""
    return [sum(a * xi for a, xi in zip(row, x)) for row in A]

def matvec_cols(A, x):
    """Output is x[0]*(column 0) + x[1]*(column 1) + ..."""
    m, n = len(A), len(A[0])
    out = [0] * m
    for j in range(n):              # for each column ...
        for i in range(m):          # ... add x[j] times that column
            out[i] += x[j] * A[i][j]
    return out

A = [[2, -1],
     [1,  1]]
x = [1, 2]
print(matvec_rows(A, x), matvec_cols(A, x))
```

```
[0, 3] [0, 3]
```

27.2.5. Reading a matrix at a glance

The column view lets you **read** a matrix geometrically. Faced with $\begin{bmatrix} 1 & 0.8 \\ 0 & 1 \end{bmatrix}$, don't see four numbers: see "$\mathbf{e_1}$ stays at $(1, 0)$; $\mathbf{e_2}$ tips over to $(0.8, 1)$". That's a horizontal shear: points slide sideways by an amount proportional to their height. Faced with $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$: "$\mathbf{e_1}$ stays, $\mathbf{e_2}$ is crushed to zero", a projection onto the x-axis.

27.2.6. Where you'd use this: many outputs from many inputs

A matrix–vector product is the right tool whenever several totals each depend on the same list of inputs.

- **Everyday: a bakery's shopping list.** A cake needs 200 g flour, 150 g sugar and 3 eggs; a batch of cookies needs 300 g flour, 100 g sugar and 1 egg. Put each product's recipe in a column. An order of 4 cakes and 2 batches is the vector $(4, 2)$, and

$$\underbrace{\begin{bmatrix} 200 & 300 \\ 150 & 100 \\ 3 & 1 \end{bmatrix}}_{\text{recipes}} \begin{bmatrix} 4 \\ 2 \end{bmatrix} = 4 \begin{bmatrix} 200 \\ 150 \\ 3 \end{bmatrix} + 2 \begin{bmatrix} 300 \\ 100 \\ 1 \end{bmatrix} = \begin{bmatrix} 1400 \\ 800 \\ 14 \end{bmatrix}$$

  is the shopping list: 1400 g flour, 800 g sugar, 14 eggs. That's the column view in action: 4 copies of the cake column plus 2 of the cookie column. Restaurant stock systems, manufacturing "bill of materials" software and diet planners all do this.

- **Science and engineering:** economists' **input–output** models (Leontief won the 1973 Nobel prize for them) are this calculation for a whole economy, with a column per industry saying what it consumes to make one unit. In electronics, the currents in a resistor network are linear in the

source voltages, so a circuit simulator stores the network as a matrix.

- **Computing:** the row view is a batch of dot products, so scoring one user against 10,000 items in a recommender, or one query against every document in a search index, is a single matrix–vector product.



Rows and columns are not interchangeable

The columns of A are the images of the basis vectors, not the rows. Writing the landing spot of $\mathbf{e}_1$ as the first **row** by mistake gives you the **transpose** (Section 27.10), which is a different transformation unless the matrix is symmetric. A rotation written this way rotates the wrong way. Some graphics APIs and textbooks use **row vectors** multiplied on the left ($\mathbf{x}A$ instead of $A\mathbf{x}$), in which case everything is transposed. DirectX conventionally uses row vectors and OpenGL column vectors; when porting code between them, this is the number-one source of bugs.

US curriculum

Matrix arithmetic (adding, multiplying, determinants and inverses of 2×2 matrices, solving systems with matrices) appears in many **Algebra II** and **Precalculus** courses, often as a computational unit (Common Core N-VM.6–12, "+" standards). Matrices **as transformations of the plane** are touched on in some Precalculus courses but are mostly taught in a first college course in **Linear Algebra**, typically taken in the first or second year of a STEM degree.

27.3. A gallery of transformations

Because columns are landing spots, you can build the matrix for any linear transformation by asking where $\mathbf{e}_1$ and $\mathbf{e}_2$ go.

27.3.1. Scaling

Stretch horizontally by s_x and vertically by s_y: $\mathbf{e}_1 \mapsto (s_x, 0)$, $\mathbf{e}_2 \mapsto (0, s_y)$.

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}.$$

A matrix that's zero off the main diagonal (top-left to bottom-right) is a **diagonal matrix**, and it just scales each axis independently. Diagonal matrices are the simplest matrices there are, and much of Chapter 28 is about turning other matrices into diagonal ones.

27.3.2. Rotation, derived from trigonometry

Rotate the plane anticlockwise by angle θ about the origin. Where does $\mathbf{e}_1 = (1, 0)$ go? It's a unit vector, so it stays on the unit circle, now at angle θ: by the definition of sine and cosine (Volume 1, Chapter 13), it lands at $(\cos\theta, \sin\theta)$.

Where does $\mathbf{e_2} = (0, 1)$ go? It starts at angle $90°$ and ends at $90° + \theta$, so it lands at $(\cos(90° + \theta), \sin(90° + \theta)) = (-\sin\theta, \cos\theta)$. (You can also see this from the diagram: the rotated $\mathbf{e_2}$ is the rotated $\mathbf{e_1}$ turned a further quarter-turn, and a quarter-turn sends (p, q) to $(-q, p)$.)

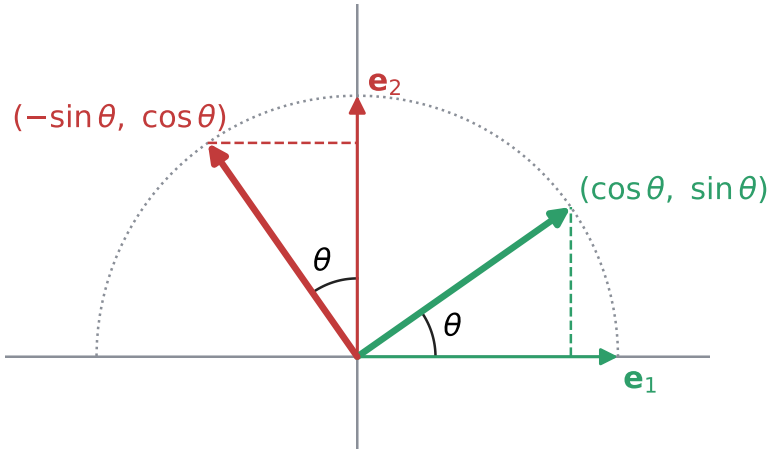


Figure 70. Rotating the basis vectors by θ: $\mathbf{e_1}$ lands at $(\cos\theta, \sin\theta)$ and $\mathbf{e_2}$ at $(-\sin\theta, \cos\theta)$. These are the columns of the rotation matrix.

Put the landing spots in as columns:

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \qquad R_\theta \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}.$$

This is the formula behind every 2D rotation in every game engine, and it falls out of nothing more than "columns are where the basis vectors go" plus the unit circle. **Worked check:** rotate $(3, 4)$ by $90°$. With $\cos 90° = 0$, $\sin 90° = 1$: $(3 \cdot 0 - 4 \cdot 1,\ 3 \cdot 1 + 4 \cdot 0) = (-4, 3)$. Its length is still 5, and $(3, 4) \cdot (-4, 3) = -12 + 12 = 0$, so it's perpendicular to the original. ✓

27.3.3. Shear, reflection, projection

- **Horizontal shear** by factor k: $\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$. Points at height y slide sideways by ky. Italic fonts are (roughly) a shear of upright ones.
- **Reflection** in the y-axis: $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$; in the line $y = x$: $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ (it swaps coordinates).
- **Projection** onto the x-axis: $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$. More generally, projection onto the line through a unit vector $\hat{\mathbf{u}} = (u_1, u_2)$ sends $\mathbf{x}$ to $(\mathbf{x} \cdot \hat{\mathbf{u}})\hat{\mathbf{u}}$ (Chapter 26); applying that to $\mathbf{e_1}$ and $\mathbf{e_2}$ gives the matrix $\begin{bmatrix} u_1^2 & u_1 u_2 \\ u_1 u_2 & u_2^2 \end{bmatrix}$.

27.3.4. Where you'd use this

- **Everyday:** a phone's photo editor implements "rotate 90°", "flip horizontal" (the reflection $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$, which is also what a selfie camera's mirror preview does) and "straighten" (a small rotation) with exactly these matrices, applied to pixel coordinates.

- **Science and engineering: robots and sensors.** A robot vacuum facing 30° anticlockwise from the room's x-axis sees an obstacle 2 m straight ahead, at $(2, 0)$ in its own frame. In room directions that's $R_{30°}(2, 0) = (2\cos 30°, 2\sin 30°) \approx (1.73, 1.00)$ m from the robot. Every robot, drone and phone (turning gyroscope readings into "which way is up") rotates measurements between frames like this. Crystallographers describe the symmetries of a crystal as a small set of rotation and reflection matrices.
- **Computing:** sprite and camera rotation in games, italic and oblique fonts (a shear), the "flatten 3D to 2D" projection in every renderer, and **data augmentation** in machine learning, where training images are randomly rotated, flipped and sheared so a classifier learns that a tilted cat is still a cat.



Non-square matrices change dimension

A 2×3 matrix has 3 columns (images of $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ in 3D), each with 2 entries (they land in 2D). So it maps $\mathbb{R}^3 \to \mathbb{R}^2$, for example by dropping the depth of a 3D point to draw it on paper: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ keeps x, y and discards z. An $m \times n$ matrix eats n-vectors and produces m-vectors. A neural-network layer from 768 to 3072 dimensions is a 3072×768 matrix.

27.4. Exercises: matrices as transformations

1. **(Recognition)** Describe in words what $\begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$, $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ do to the plane.
2. **(Application)** Write the matrix that sends $\mathbf{e}_1$ to $(1, 2)$ and $\mathbf{e}_2$ to $(3, -1)$, and use it to transform $(2, 1)$ by both the row and column methods.
3. **(Application)** Find the matrix that rotates by $60°$, and apply it to $(2, 0)$.
4. **(Problem solving)** Is $T(x, y) = (x + 1, y)$ linear? Is $T(x, y) = (y, x^2)$? Justify each with the linearity rules.

▼ **Answers**

1. Uniform scaling by 3 (everything three times as far from the origin); anticlockwise rotation by $90°$ ($\mathbf{e}_1 \mapsto \mathbf{e}_2$, $\mathbf{e}_2 \mapsto -\mathbf{e}_1$); reflection in the x-axis (flip vertically).
2. $A = \begin{bmatrix} 1 & 3 \\ 2 & -1 \end{bmatrix}$. Rows: $(1 \cdot 2 + 3 \cdot 1,\ 2 \cdot 2 + (-1) \cdot 1) = (5, 3)$. Columns: $2(1, 2) + 1(3, -1) = (2 + 3, 4 - 1) = (5, 3)$.
3. $\cos 60° = \frac{1}{2}, \sin 60° = \frac{\sqrt{3}}{2}$: $R = \begin{bmatrix} 1/2 & -\sqrt{3}/2 \\ \sqrt{3}/2 & 1/2 \end{bmatrix}$. $R(2, 0) = (1, \sqrt{3}) \approx (1, 1.732)$, length still 2.
4. $(x + 1, y)$ is not linear: $T(0, 0) = (1, 0) \neq \mathbf{0}$ (it's a translation). (y, x^2) is not linear: $T(2 \cdot (1, 0)) = T(2, 0) = (0, 4)$ but $2T(1, 0) = 2(0, 1) = (0, 2)$.

27.5. Matrix multiplication is composition

27.5.1. The problem

A game rotates a sprite by $90°$ and then shears it. With a million vertices, you'd rather not apply two matrices to every vertex; you'd like **one** matrix that does both. What is it? And does the order matter?

27.5.2. Intuition: follow the basis vectors again

Let B be the first transformation and A the second. The combined transformation sends $\mathbf{x}$ to $A(B\mathbf{x})$. Doing one linear transformation after another is still linear (grid lines stay straight through both steps), so it has a matrix, and we find it as always: by tracking the basis vectors.

- B sends $\mathbf{e_1}$ to B's first column, call it $\mathbf{b_1}$. Then A sends that to $A\mathbf{b_1}$.
- Likewise $\mathbf{e_2}$ ends up at $A\mathbf{b_2}$.

So the combined matrix, written AB, has columns $A\mathbf{b_1}$ and $A\mathbf{b_2}$:

$$AB = A\Big[\,\mathbf{b_1}\;\Big|\;\mathbf{b_2}\,\Big] = \Big[\,A\mathbf{b_1}\;\Big|\;A\mathbf{b_2}\,\Big].$$

Each column of the product is A times the corresponding column of B. That's the definition of matrix multiplication, and it is **forced** by the requirement that $(AB)\mathbf{x} = A(B\mathbf{x})$.

Now apply the row view of matrix–vector multiplication to each $A\mathbf{b}_j$: entry i of column j is (row i of A) · (column j of B). That's the familiar "row times column" rule:

$$(AB)_{ij} = \sum_k a_{ik}\, b_{kj}.$$



Read products right to left

In $AB\mathbf{x}$, the transformation **nearest the vector acts first**. AB means "do B, then A ", exactly like function composition $f(g(x))$ or $(f \circ g)(x)$, where g happens first (Volume 1, Chapter 11). Graphics code that builds $M = TRS$ (translate · rotate · scale) applies the scale first and the translation last.

27.5.3. Order matters: matrix multiplication is not commutative

Let $R = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ (rotate $90°$) and $S = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ (shear). Compute both orders.

SR (rotate first, then shear). Columns of R are $(0,1)$ and $(-1,0)$. Apply S to each:

$$S\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\cdot 0 + 1\cdot 1\\ 0\cdot 0 + 1\cdot 1\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}, \qquad S\begin{bmatrix}-1\\0\end{bmatrix} = \begin{bmatrix}-1+0\\0+0\end{bmatrix} = \begin{bmatrix}-1\\0\end{bmatrix}, \qquad SR = \begin{bmatrix}1 & -1\\1 & 0\end{bmatrix}.$$

RS (shear first, then rotate). Columns of S are $(1,0)$ and $(1,1)$:

$$R\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}, \qquad R\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0-1\\1+0\end{bmatrix} = \begin{bmatrix}-1\\1\end{bmatrix}, \qquad RS = \begin{bmatrix}0 & -1\\1 & 1\end{bmatrix}.$$

$SR \neq RS$. The picture makes it obvious:

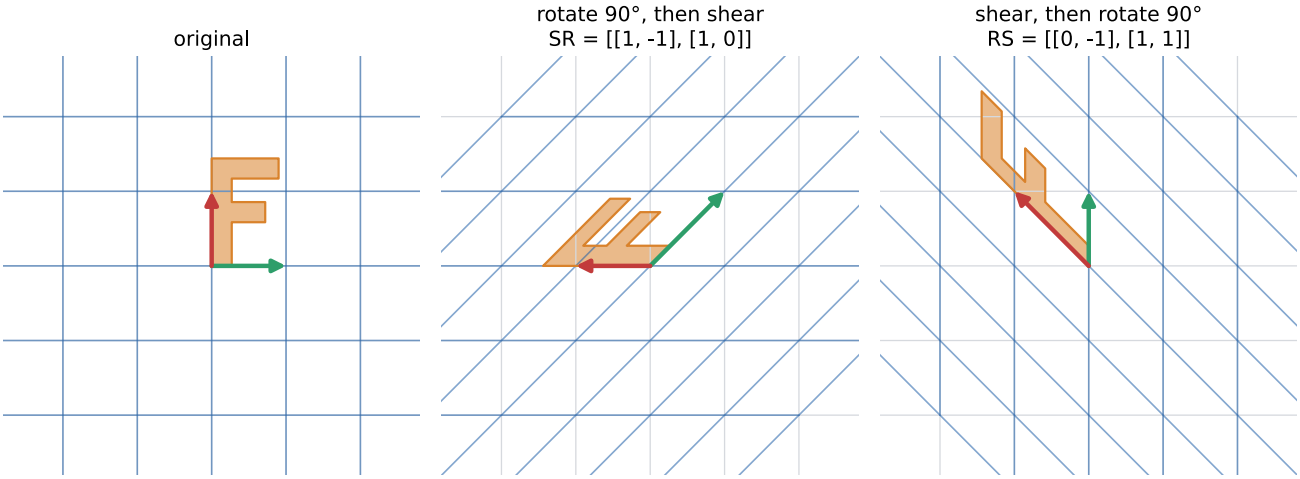


Figure 71. The same two transformations in different orders give different results. Matrix multiplication is not commutative.

Think of everyday actions: "put on socks, then shoes" differs from "shoes, then socks". Some pairs do commute (two rotations about the same point; two scalings; anything with the identity), but you may never **assume** it.

Example 33. Procedure — multiply two matrices

Input: an $m \times n$ matrix A and an $n \times p$ matrix B.

Steps:

1. Check the **inner dimensions** match: columns of A (n) = rows of B (n). The result will be $m \times p$ (the **outer** dimensions).
2. For each row i of A and each column j of B: compute the dot product $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$.
3. Place c_{ij} in row i, column j of the result.

Output: $C = AB$, which represents "apply B, then A". Cost: $m \cdot n \cdot p$ multiplications.

Try it: Multiply row by column, stepped .

27.5.4. Worked example

Compute $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$. *Try it:* Step through this product .

$$c_{11} = (\text{row } 1) \cdot (\text{col } 1) = 1 \cdot 5 + 2 \cdot 7 = 19, \qquad c_{12} = (\text{row } 1) \cdot (\text{col } 2) = 1 \cdot 6 + 2 \cdot 8 = 22,$$
$$c_{21} = (\text{row } 2) \cdot (\text{col } 1) = 3 \cdot 5 + 4 \cdot 7 = 43, \qquad c_{22} = (\text{row } 2) \cdot (\text{col } 2) = 3 \cdot 6 + 4 \cdot 8 = 50,$$

so the product is $\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$. (In the other order you'd get $\begin{bmatrix} 23 & 34 \\ 31 & 46 \end{bmatrix}$: different again.) *Try it:* Try the other order .

27.5.5. Identity, associativity, and powers

The **identity matrix** I has 1s on the diagonal and 0s elsewhere. It sends every basis vector to itself, so it's the "do nothing" transformation: $AI = IA = A$. It plays the role of 1 in ordinary multiplication.

Matrix multiplication **is associative**: $(AB)C = A(BC)$. Both sides mean "do C, then B, then A", so they must be the same transformation. This is why a graphics engine can pre-multiply a chain of a dozen transformations into a single matrix once per frame and then apply that one matrix to every vertex. It's also why A^n (A multiplied by itself n times, "apply A n times") is unambiguous, which Chapter 28 exploits heavily.



Identity matrix and matrix product

- The **identity matrix** I_n is the $n \times n$ matrix with 1s on the main diagonal and 0s elsewhere. $I\mathbf{x} = \mathbf{x}$ for every $\mathbf{x}$.
- The **product** AB is defined when (columns of A) = (rows of B), and $(AB)_{ij}$ = (row i of A) · (column j of B). It represents applying B then A.
- Laws: associative $(AB)C = A(BC)$; distributive $A(B + C) = AB + AC$; **not** commutative in general.

27.5.6. Where you'd use this

- **Everyday: price a recipe once, not every time.** Continue the bakery: flour costs £1.20/kg, sugar £2/kg and eggs 25p each, so the price row vector is $\mathbf{p} = (0.0012, 0.002, 0.25)$ (per gram, per gram, per egg). The product $\mathbf{p}R$ (a 1×3 times a 3×2) gives the ingredient cost of each product: $0.24 + 0.30 + 0.75 = 1.29$, so £1.29 per cake, and $0.36 + 0.20 + 0.25 = 0.81$, so £0.81 per batch of cookies. Associativity says $(\mathbf{p}R)\mathbf{q} = \mathbf{p}(R\mathbf{q})$: costing the order of 4 cakes and 2 batches either way gives $4(1.29) + 2(0.81) = 6.78$, that is £6.78.
- **Science and engineering:** a robot arm's hand position is found by multiplying one transformation matrix per joint (shoulder, elbow, wrist), which is called **forward kinematics**. In optics, each lens and gap in a camera or telescope is a 2×2 "ray transfer" matrix, and the whole instrument is their product.
- **Computing:** a game object inside a vehicle inside a world has its matrix computed as a product down the **scene graph**; animation systems compose bone transformations the same way; and Chapter 28 uses powers A^n to step a Markov chain or a Fibonacci recurrence many steps at once.



Rotations add, and trigonometry falls out

Rotating by β and then by α is obviously rotating by $\alpha + \beta$. So $R_\alpha R_\beta = R_{\alpha+\beta}$. Multiply out the left side:

$$\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\beta - \sin\alpha\sin\beta & \cdots \\ \sin\alpha\cos\beta + \cos\alpha\sin\beta & \cdots \end{bmatrix}.$$

Compare with the first column of $R_{\alpha+\beta}$, which is $(\cos(\alpha+\beta), \sin(\alpha+\beta))$. You've just derived both angle-addition formulas from Volume 1, Chapter 13 , the ones that took a page of geometry to prove at school, from one matrix multiplication.

27.5.7. Implementing it, and what it costs

PYTHON

```
import numpy as np

def matmul(A, B):
    """Product of an m x n and an n x p matrix (lists of rows)."""
    m, n, p = len(A), len(B), len(B[0])
    assert len(A[0]) == n, "inner dimensions must match"
    C = [[0] * p for _ in range(m)]
    for i in range(m):              # each row of A
        for j in range(p):          # each column of B
            for k in range(n):      # dot product of row i and column j
                C[i][j] += A[i][k] * B[k][j]
    return C

R = [[0, -1], [1, 0]]               # rotate 90 degrees
S = [[1, 1], [0, 1]]                # shear
print(matmul(S, R))                 # rotate first, then shear
print(matmul(R, S))                 # shear first, then rotate

# NumPy agrees, and is enormously faster for big matrices.
print(np.array(S) @ np.array(R))
```

```
[[1, -1], [1, 0]]
[[0, -1], [1, 1]]
[[ 1 -1]
 [ 1  0]]
```

Three nested loops of length n mean n^3 multiplications for two $n \times n$ matrices: $O(n^3)$. Doubling n multiplies the work by 8. For $n = 1000$ that's a billion multiply-adds: a minute or more in pure Python, a few tens of milliseconds in NumPy.

Computer-science connection: why matrix multiplication is the most optimised code on Earth

NumPy's `@` calls a **BLAS** library (OpenBLAS, Intel MKL, Apple Accelerate) whose `GEMM` ("general matrix multiply") routine is hand-tuned in assembly. It still does $O(n^3)$ arithmetic, but it reorganises the loops into **blocks** that fit in the CPU caches, uses SIMD instructions, and runs on every core, reaching a large fraction of the hardware's peak speed. The naive triple loop spends most of its time waiting for memory. On a GPU, the same operation runs on thousands of cores, and modern GPUs have dedicated **tensor cores** that do nothing but small matrix multiplies. Training a large neural network is, to a first approximation, a few trillion trillion multiply-adds inside matrix multiplications.

Can you beat n^3? Yes, in theory. Volker Strassen showed in 1969 that two 2×2 block matrices can be multiplied with 7 block multiplications instead of 8, giving $O(n^{\log_2 7}) \approx O(n^{2.81})$. Research has pushed the exponent below 2.372, but those algorithms only win for astronomically large matrices. Strassen itself is occasionally used in practice; the rest are "galactic algorithms". The connection to Volume 1, Chapter 12 : the exponent $\log_2 7$ comes from a divide-and-conquer recurrence.

27.6. The determinant: how much space changes

27.6.1. The problem

When a matrix transforms the plane, the unit square (the square spanned by $\mathbf{e}_1$ and $\mathbf{e}_2$) becomes a parallelogram spanned by the two columns. How does its area change? And can we tell from the numbers alone whether the transformation flips space over, or flattens it completely?

27.6.2. Intuition

Because grid lines stay parallel and evenly spaced, **every** grid square becomes a copy of the same parallelogram. So every region's area is multiplied by the same factor: the area of the transformed unit square. That factor, with a sign attached to record whether the transformation flips orientation, is the **determinant**, $\det A$.

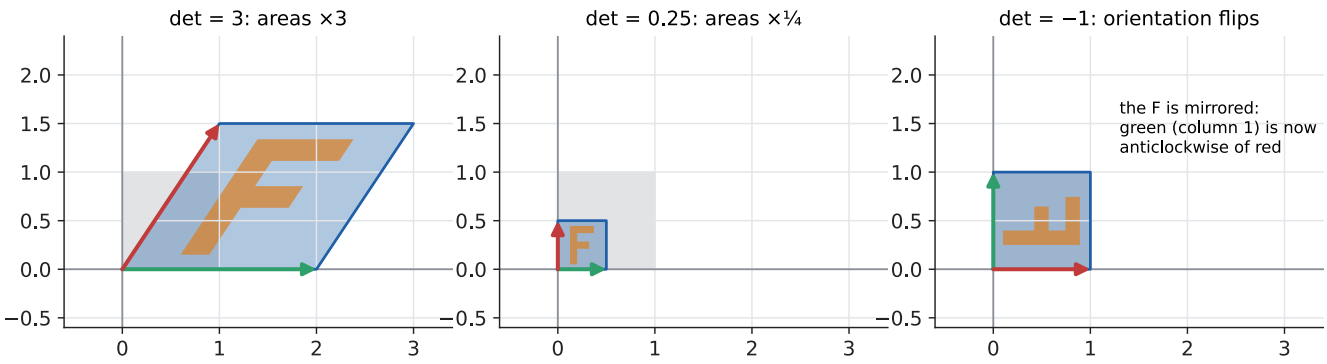


Figure 72. The determinant is the factor by which areas are multiplied (blue parallelogram versus grey unit square). A negative determinant means orientation is reversed: the F is mirrored.

27.6.3. Why it works: deriving the formula

Let $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ with all entries positive and the columns (a, c) and (b, d) arranged as in the first panel above. The parallelogram fits in a rectangle of width $a + b$ and height $c + d$. The parts of the rectangle outside the parallelogram are two triangles of area $\frac{1}{2}ac$, two of area $\frac{1}{2}bd$, and two small rectangles of area bc. Subtract them:

$$\begin{aligned} \text{area} &= (a+b)(c+d) - 2 \cdot \tfrac{1}{2}ac - 2 \cdot \tfrac{1}{2}bd - 2bc \\ &= ac + ad + bc + bd - ac - bd - 2bc && \text{(expand)} \\ &= ad - bc. && \text{(collect)} \end{aligned}$$

(Sketching that rectangle and its six pieces on paper is well worth the minute it takes.) In general,

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc,$$

and the sign comes out right automatically: swapping the columns (which mirrors the picture) gives $bc - ad$, the negative. Notice that for the cross-product-style "2D cross" $a_1b_2 - a_2b_1$ from Chapter 26, this is the same expression: its sign said whether the second vector was anticlockwise of the first, which is exactly orientation.



Determinant

The **determinant** of a square matrix, written $\det A$ or $|A|$, is the signed factor by which the transformation scales area (2D) or volume (3D and up).

- $|\det A|$ = scale factor for area/volume.
- $\det A < 0$: orientation is reversed (a mirror image).
- $\det A = 0$: space is flattened into a lower dimension; the matrix is **singular**.
- For 2×2: $\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$.
- $\det(AB) = \det A \cdot \det B$ (scale factors multiply when you compose) and $\det I = 1$.

27.6.4. Worked example

For the three panels above:

$$\det\begin{bmatrix}2 & 1\\ 0 & 1.5\end{bmatrix} = 2(1.5) - 1(0) = 3,$$
$$\det\begin{bmatrix}0.5 & 0\\ 0 & 0.5\end{bmatrix} = 0.25 - 0 = 0.25,$$
$$\det\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} = 0 - 1 = -1.$$

A rotation has $\det R_\theta = \cos^2\theta + \sin^2\theta = 1$: it neither grows nor shrinks area and doesn't mirror. A shear $\begin{bmatrix}1 & k\\ 0 & 1\end{bmatrix}$ has determinant 1 too: it slants squares into parallelograms of the same base and height. Scaling a 2D shape by 3 in both directions multiplies its area by $\det(3I) = 9$, not 3.

27.6.5. Three dimensions, briefly

For a 3×3 matrix, the determinant is the signed **volume** of the parallelepiped (slanted box) spanned by the three columns. One way to compute it is **cofactor expansion** along the first row:

$$\det\begin{bmatrix}a_1 & a_2 & a_3\\ b_1 & b_2 & b_3\\ c_1 & c_2 & c_3\end{bmatrix} = a_1(b_2c_3 - b_3c_2) - a_2(b_1c_3 - b_3c_1) + a_3(b_1c_2 - b_2c_1).$$

Each bracket is a 2×2 determinant of what's left after deleting row 1 and the entry's column; signs alternate $+, -, +$. If you put $\mathbf{i}, \mathbf{j}, \mathbf{k}$ in the first row and two vectors in the others, this formula produces exactly the cross product of Chapter 26, which is the mnemonic promised there. For anything larger than 3×3, don't expand by hand (the work grows like $n!$); computers use elimination, as described next.

27.6.6. Where you'd use this

- **Everyday and surveying: area from coordinates.** A triangular plot of land has corners $(0, 0)$, $(40, 10)$ and $(15, 30)$ in metres. Its two edge vectors from the first corner are the columns of a matrix, and the triangle is half the parallelogram they span:

  $$\text{area} = \tfrac{1}{2}\left|\det\begin{bmatrix}40 & 15\\ 10 & 30\end{bmatrix}\right| = \tfrac{1}{2}\,|40 \cdot 30 - 15 \cdot 10| = \tfrac{1}{2}(1200 - 150) = 525\ \text{m}^2.$$

  Splitting any polygon into triangles this way gives the **shoelace formula**, which surveyors, GIS software and mapping apps use to compute the area of a field or a building footprint from its corner coordinates.

- **Science:** when you change variables in a multiple integral (to polar coordinates, say), the determinant of the Jacobian matrix is the local area-scaling factor (Chapter 20). In mechanics, a negative determinant is how you spot that a coordinate change has mirrored space (switched right- and left-handed axes).

- **Computing:** the sign of a 2×2 determinant is the standard **orientation test** in computational geometry: is point C to the left or right of the line from A to B? Convex-hull algorithms, polygon triangulation, and "did these two line segments cross?" checks in games and CAD all reduce to it.



The determinant is not a measure of "how invertible"

$\det A = 0$ exactly means singular. But a **small** determinant does not mean "nearly singular". The matrix $0.1\, I_{100}$ (shrink every one of 100 axes by 10) has determinant 10^{-100}, yet it's perfectly well-behaved and trivially inverted. And a matrix with determinant 1 can be extremely close to singular, like $\begin{bmatrix} 10^{6} & 0 \\ 0 & 10^{-6} \end{bmatrix}$. To judge how close a matrix is to singular, numerical analysts use the **condition number** (Chapter 32), which compares the largest and smallest stretching factors. And in floating point, a computed determinant of 10^{-17} is almost always "zero plus rounding error", never test `det == 0`.

27.7. Inverses and singular matrices

27.7.1. The problem

A camera transformation maps world coordinates to screen coordinates. The player clicks a pixel: which point in the world did they click? An image has been deliberately rotated and sheared; how do you put it back? Both ask: **can a transformation be undone, and if so, by what matrix?**

27.7.2. Intuition: undo, if nothing was lost

The **inverse** of A, written A^{-1}, is the transformation that puts everything back: $A^{-1}A = I$, and also $AA^{-1} = I$. For a rotation by θ it's rotation by $-\theta$. For a scaling by 2 it's scaling by $\frac{1}{2}$. For a shear by k it's the shear by $-k$.

But some transformations cannot be undone. Projection onto the x-axis sends $(3, 5)$ and $(3, -2)$ both to $(3, 0)$. Given only $(3, 0)$, there's no way to know where it came from: information has been destroyed. This happens exactly when the transformation squashes space into a lower dimension, which is exactly when the determinant is zero.

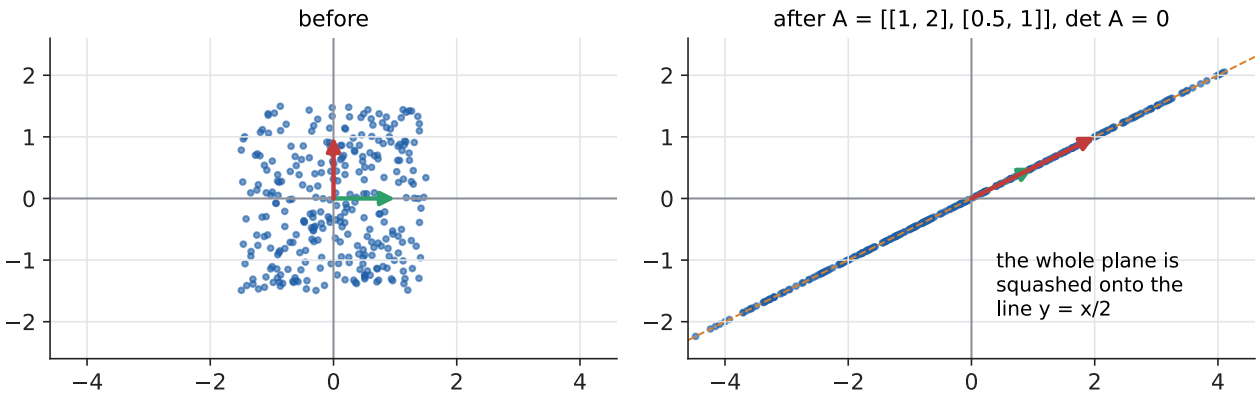


Figure 73. A singular matrix flattens the plane. Every point lands on one line, and many different points land on the same spot, so no inverse can exist.

In that example the second column $(2, 1)$ is exactly twice the first column $(1, 0.5)$. Both basis vectors land on the same line, so their combinations (everything) land on that line. **Singular means the columns are linearly dependent** (Chapter 26): they fail to span the whole space.



Invertible and singular

A square matrix A is **invertible** (or **non-singular**) if there's a matrix A^{-1} with $AA^{-1} = A^{-1}A = I$. Otherwise it is **singular**. For a square matrix, all of these are equivalent: A is invertible; $\det A \neq 0$; the columns are linearly independent; the columns span the whole space; $A\mathbf{x} = \mathbf{b}$ has exactly one solution for every $\mathbf{b}$; $A\mathbf{x} = \mathbf{0}$ only for $\mathbf{x} = \mathbf{0}$. Non-square matrices never have a two-sided inverse.

27.7.3. Deriving the 2×2 inverse

Undoing A means: given an output (p, q), find the input (x, y) with

$$\begin{aligned} ax + by &= p \\ cx + dy &= q. \end{aligned}$$

That's a linear system, so solve it by elimination (Volume 1, Chapter 6). To eliminate y, multiply the first equation by d and the second by b, and subtract:

$$\begin{array}{rcl} adx + bdy & = & dp \\ bcx + bdy & = & bq \\ \hline (ad - bc)\,x & = & dp - bq. \end{array}$$

To eliminate x, multiply the first by c, the second by a, and subtract the first from the second: $(ad - bc)\,y = aq - cp$. If $ad - bc \neq 0$, divide:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} dp - bq \\ -cp + aq \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix}.$$

The matrix taking (p, q) back to (x, y) is the inverse:

$$\boxed{\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}}$$

Swap the diagonal, negate the off-diagonal, divide by the determinant. And there's the determinant, appearing on its own: if it's zero, you'd divide by zero, which is the algebra's way of saying "this can't be undone".

Example 34. Procedure — inverting a 2×2 matrix

Input: $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

Steps:

1. Compute $\det A = ad - bc$. If it is 0 (in floating point: tiny relative to the entries), stop: A is singular.
2. Swap a and d; change the signs of b and c.
3. Multiply every entry by $1/\det A$.
4. Check: AA^{-1} should be I.

Output: A^{-1}.

27.7.4. Worked example

Invert $A = \begin{bmatrix} 3 & 1 \\ 5 & 2 \end{bmatrix}$ and use it to solve $3x + y = 5, 5x + 2y = 9$. *Try it:* Check this inverse ; Solve the system .

$$\begin{aligned} \det A &= 3 \cdot 2 - 1 \cdot 5 = 1, \\ A^{-1} &= \frac{1}{1}\begin{bmatrix} 2 & -1 \\ -5 & 3 \end{bmatrix}, \\ \text{check: } AA^{-1} &= \begin{bmatrix} 3\cdot 2 + 1\cdot(-5) & 3\cdot(-1) + 1\cdot 3 \\ 5\cdot 2 + 2\cdot(-5) & 5\cdot(-1) + 2\cdot 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \checkmark \\ \begin{bmatrix} x \\ y \end{bmatrix} &= A^{-1}\begin{bmatrix} 5 \\ 9 \end{bmatrix} = \begin{bmatrix} 2\cdot 5 - 1\cdot 9 \\ -5\cdot 5 + 3\cdot 9 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}. \end{aligned}$$

Check in the original equations: $3 + 2 = 5$ ✓, $5 + 4 = 9$ ✓.

```python
def inverse_2x2(A, tol=1e-12):
    (a, b), (c, d) = A
    det = a * d - b * c
    if abs(det) < tol:
        raise ValueError("matrix is singular (or nearly so): no inverse")
    return [[ d / det, -b / det],
            [-c / det,  a / det]]

A = [[3, 1], [5, 2]]
print(inverse_2x2(A))
```

```
[[2.0, -1.0], [-5.0, 3.0]]
```

27.7.5. Where you'd use this

- **Everyday: recovering hidden quantities.** A café sold 120 drinks yesterday, coffees at £3 and teas at £2, and took £310. With x coffees and y teas, $\begin{bmatrix}1 & 1\\ 3 & 2\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}120\\ 310\end{bmatrix}$. The determinant is $2-3=-1$, so the inverse is $\frac{1}{-1}\begin{bmatrix}2 & -1\\ -3 & 1\end{bmatrix}=\begin{bmatrix}-2 & 1\\ 3 & -1\end{bmatrix}$, giving $x=-240+310=70$ coffees and $y=360-310=50$ teas. (Check: $70+50=120$, $210+100=310$.) *Try it:* Solve the café problem .
- **Science and engineering:** GPS receivers, circuit simulators and structural-engineering software all solve linear systems, from a handful of unknowns to millions. Undoing a known distortion is an inverse too: a scanner that knows how its optics warp the page applies the inverse map to straighten the image.
- **Computing:** when you click on a 3D scene, the game must work out which point in the world you clicked. The renderer maps world to screen with a matrix M; **picking** applies M^{-1} to the click. Because rotations are orthogonal, their inverse is just the transpose (Section 27.10), which is why engines keep rotation, scale and translation separate: each is trivial to invert on its own.



Socks and shoes

The inverse of a product reverses the order: $(AB)^{-1}=B^{-1}A^{-1}$. To undo "socks, then shoes" you remove shoes first, then socks. Check: $(AB)(B^{-1}A^{-1})=A(BB^{-1})A^{-1}=AIA^{-1}=AA^{-1}=I$. Also $(A^{-1})^{-1}=A$ and $\det(A^{-1})=1/\det A$ (if A multiplies areas by 3, undoing it divides them by 3).

27.8. Solving $Ax=b$

27.8.1. The problem, recast

Any system of linear equations can be written as one matrix equation. The system

$$\begin{aligned} 2x + y - z &= 1 \\ x + 3y + 2z &= 13 \\ x &= 2 \end{aligned} \qquad \text{is} \qquad \underbrace{\begin{bmatrix} 2 & 1 & -1 \\ 1 & 3 & 2 \\ 1 & 0 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 1 \\ 13 \\ 2 \end{bmatrix}}_{\mathbf{b}}.$$

Check by the row view: row 1 of A dotted with $\mathbf{x}$ is $2x + y - z$, the first left-hand side. This compact form is how systems are written in every linear-algebra paper and every numerical library.

27.8.2. Two pictures of the same equation

Volume 1, Chapter 6 taught you the **row picture**: each equation is a line (or plane), and the solution is where they all meet. Matrices add the **column picture**: $A\mathbf{x}$ is a combination of the columns of A, so solving $A\mathbf{x} = \mathbf{b}$ asks

> *"How much of each column do I need to build $\mathbf{b}$?*

and, as a transformation: **which input does A send to $\mathbf{b}$?** That gives a clean geometric account of the three cases from Volume 1, Chapter 6 :

- A invertible: A reaches every point exactly once, so there's exactly one solution, $\mathbf{x} = A^{-1}\mathbf{b}$.
- A singular and $\mathbf{b}$ **off** the squashed image (not in the span of the columns): no input reaches $\mathbf{b}$; no solution.
- A singular and $\mathbf{b}$ **on** the image: many inputs were squashed onto $\mathbf{b}$; infinitely many solutions.

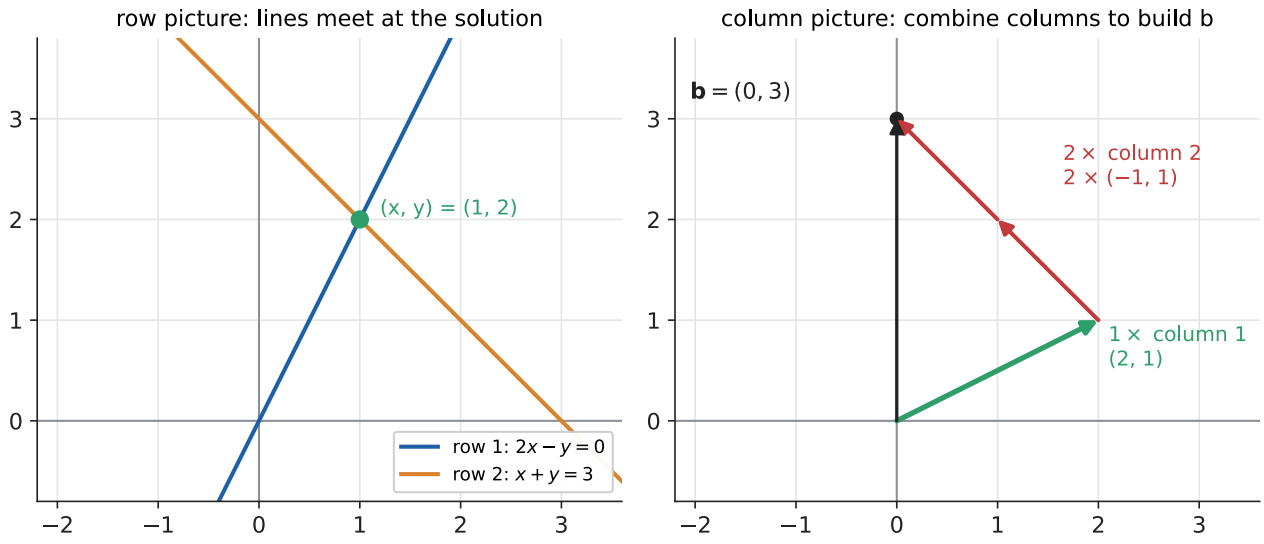


Figure 74. Two pictures of the system $2x - y = 0$, $x + y = 3$, which is $A\mathbf{x} = \mathbf{b}$ with $A = \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}$ and $\mathbf{b} = (0, 3)$. Left: each row is a line, and the solution $(1, 2)$ is where they cross. Right: the same answer read as "1 of column 1 plus 2 of column 2 builds $\mathbf{b}$", exactly the landing of $\mathbf{v} = (1, 2)$ in the columns diagram earlier.

27.8.3. Rank: how many dimensions survive

The **rank** of a matrix is the dimension of its output: the number of independent directions the columns span. A 3×3 matrix of rank 3 fills all of space (invertible); rank 2 flattens space onto a plane; rank 1 crushes it onto a line; rank 0 is the zero matrix. The singular example above has rank 1. Rank is the

honest measure of how much information a transformation preserves, and it returns in Chapter 28 as the key to compressing data with low-rank approximations.



Rank

The **rank** of a matrix is the number of linearly independent columns (equivalently, of independent rows: the two counts always agree), which is the dimension of the set of all outputs $A\mathbf{x}$ (the **column space** or **image**). An $n \times n$ matrix is invertible exactly when its rank is n (**full rank**). The input directions squashed to zero form the **null space** (or **kernel**); its dimension is $n - \text{rank}$.

27.8.4. Choosing a method

PYTHON

```python
import numpy as np

A = np.array([[2.0, 1.0, -1.0],
              [1.0, 3.0,  2.0],
              [1.0, 0.0,  0.0]])
b = np.array([1.0, 13.0, 2.0])
x = np.linalg.solve(A, b)
print(x, np.allclose(A @ x, b))

print(round(np.linalg.det(A), 6), np.linalg.matrix_rank(A))
S = np.array([[1.0, 2.0], [0.5, 1.0]])
print(np.linalg.det(S), np.linalg.matrix_rank(S))
```

```
[2. 1. 4.] True
5.0 3
0.0 1
```

- **By hand, 2×2:** use the inverse formula or elimination; both are quick.
- **By hand, 3×3:** use elimination (Volume 1, Chapter 6). The inverse is much more work.
- **In code, any size:** call `np.linalg.solve(A, b)`. It runs Gaussian elimination (as an **LU factorisation**) in $O(n^3)$.
- **Many right-hand sides with the same A:** factorise once (`scipy.linalg.lu_factor`), then each solve is only $O(n^2)$.



Don't compute the inverse to solve a system

$\mathbf{x} = A^{-1}\mathbf{b}$ is the right way to **think**, and the wrong way to **compute**. `np.linalg.inv(A) @ b` does roughly three times the work of `np.linalg.solve(A, b)` and is less accurate in floating point. Numerical analysts have a saying: "if you're computing an inverse, you're probably doing something wrong". Keep A^{-1} in your formulas and `solve` in your code.

27.9. Exercises: products, determinants, inverses

1. **(Recognition)** Which products are defined, and what shape are they: $(2 \times 3)(3 \times 4)$, $(3 \times 2)(3 \times 2)$, $(1 \times 5)(5 \times 1)$, $(5 \times 1)(1 \times 5)$?
2. **(Application)** Compute $\begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} 1 & -1 \\ 4 & 2 \end{bmatrix}$.
3. **(Application)** Find the determinant and inverse of $\begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix}$.
4. **(Application)** For which value of k is $\begin{bmatrix} 2 & k \\ 3 & 6 \end{bmatrix}$ singular? Describe its image for that k.
5. **(Problem solving)** A transformation doubles all areas and preserves orientation; another triples areas and reverses orientation. What is the determinant of "the first, then the second"? Of its inverse?

▼ **Answers**

1. 2×4; undefined (inner $2 \neq 3$); 1×1 (a single number: it's a dot product); 5×5 (an **outer product**, every entry of one times every entry of the other).
2. Row 1: $(2 \cdot 1 + 0 \cdot 4,\ 2 \cdot (-1) + 0 \cdot 2) = (2, -2)$. Row 2: $(1 \cdot 1 + 3 \cdot 4,\ 1 \cdot (-1) + 3 \cdot 2) = (13, 5)$. Result $\begin{bmatrix} 2 & -2 \\ 13 & 5 \end{bmatrix}$.
3. $\det = 24 - 14 = 10$. Inverse $\frac{1}{10}\begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} 0.6 & -0.7 \\ -0.2 & 0.4 \end{bmatrix}$. Check the top-left of the product: $4(0.6) + 7(-0.2) = 2.4 - 1.4 = 1$. ✓
4. $\det = 12 - 3k = 0$ gives $k = 4$. Then the columns $(2, 3)$ and $(4, 6)$ are parallel (the second is twice the first), so the image is the line through the origin in direction $(2, 3)$, i.e. $y = \frac{3}{2}x$.
5. Determinants multiply: $2 \times (-3) = -6$. The inverse has determinant $-\frac{1}{6}$.

27.10. The transpose

27.10.1. Definition and meaning

The **transpose** A^{T} of a matrix flips it across its main diagonal: rows become columns. If A is $m \times n$, then A^{T} is $n \times m$, with $(A^{\mathsf{T}})_{ij} = a_{ji}$.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^{\mathsf{T}} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}.$$

The transpose has less obvious geometry than the other operations, but it's everywhere in practice, for three reasons.

1. It turns dot products into matrix products. Treat a column vector as an $n \times 1$ matrix. Then $\mathbf{a}^{\mathsf{T}}$ is $1 \times n$, and $\mathbf{a}^{\mathsf{T}}\mathbf{b}$ is $(1 \times n)(n \times 1) = 1 \times 1$: a single number, which is exactly $\mathbf{a} \cdot \mathbf{b}$. That's why papers write $\mathbf{q}^{\mathsf{T}}\mathbf{d}$ for the dot product, and $\mathbf{x}^{\mathsf{T}}A\mathbf{x}$ for "transform $\mathbf{x}$, then dot it with itself".

2. It reverses products: $(AB)^\mathsf{T} = B^\mathsf{T}A^\mathsf{T}$, the same socks-and-shoes reversal as the inverse.

3. For rotations, it is the inverse. The columns of a rotation matrix are perpendicular unit vectors ($\mathbf{e}_1$ and $\mathbf{e}_2$ rotated together). In $R^\mathsf{T}R$, entry (i, j) is (column i) · (column j), which is 1 when $i = j$ and 0 otherwise. So $R^\mathsf{T}R = I$ and $R^{-1} = R^\mathsf{T}$: inverting a rotation costs nothing. Matrices with orthonormal columns are called **orthogonal matrices**; they are the rigid motions (rotations and reflections) that preserve all lengths and angles.



Transpose, symmetric, orthogonal

- **Transpose** A^T: rows and columns swapped. $(A^\mathsf{T})^\mathsf{T} = A$, $(AB)^\mathsf{T} = B^\mathsf{T}A^\mathsf{T}$, $\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^\mathsf{T}\mathbf{b}$. Other notations: A^T, A' (statistics, MATLAB), `A.T` (NumPy).
- **Symmetric matrix**: $A^\mathsf{T} = A$ (a mirror image across the diagonal). Covariance matrices and distance matrices are symmetric; Chapter 28 shows they have especially beautiful eigenvectors.
- **Orthogonal matrix**: $Q^\mathsf{T}Q = I$, so $Q^{-1} = Q^\mathsf{T}$. Its columns are orthonormal; it preserves lengths and angles; $\det Q = \pm 1$.

27.11. Translation and homogeneous coordinates

27.11.1. The problem

Rotating a sprite about its own centre, rather than about the origin, requires moving it to the origin, rotating, and moving it back. But moving (translation) isn't linear: it doesn't fix the origin, so **no** 2×2 matrix can do it. That spoils the plan of combining every transformation into one matrix.

27.11.2. The trick: add a dimension

Write the 2D point (x, y) as the 3D vector $(x, y, 1)$. Now consider the 3×3 matrix

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}, \qquad T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}.$$

The constant 1 in the third slot, multiplied by the third column, smuggles in the shift. In three dimensions, translation is a **shear** that slides the plane $z = 1$ sideways, and shears are linear. Rotations and scalings embed by putting their 2×2 matrix in the top-left corner with a 1 in the bottom-right. Now **everything**, including translation, is a matrix, and any sequence of moves collapses into one matrix by multiplication.

This also encodes the point/vector distinction from Chapter 26. A **point** gets a final coordinate 1 and is translated. A **direction** vector gets a final coordinate 0, so translation leaves it alone, which is correct: moving a spaceship doesn't change which way "forward" is. Subtracting two points (1 − 1 = 0) gives a vector, automatically.



Homogeneous coordinates

Homogeneous coordinates represent a 2D point (x, y) as $(x, y, 1)$ and a 3D point (x, y, z) as $(x, y, z, 1)$, so that translations, rotations, scalings and (in 3D graphics) perspective projections are all matrix multiplications: 3×3 matrices for 2D graphics, 4×4 for 3D. Directions use a final coordinate of 0. More generally (X, Y, W) with $W \neq 0$ represents the point $(X/W, Y/W)$; dividing by W is what makes perspective work.

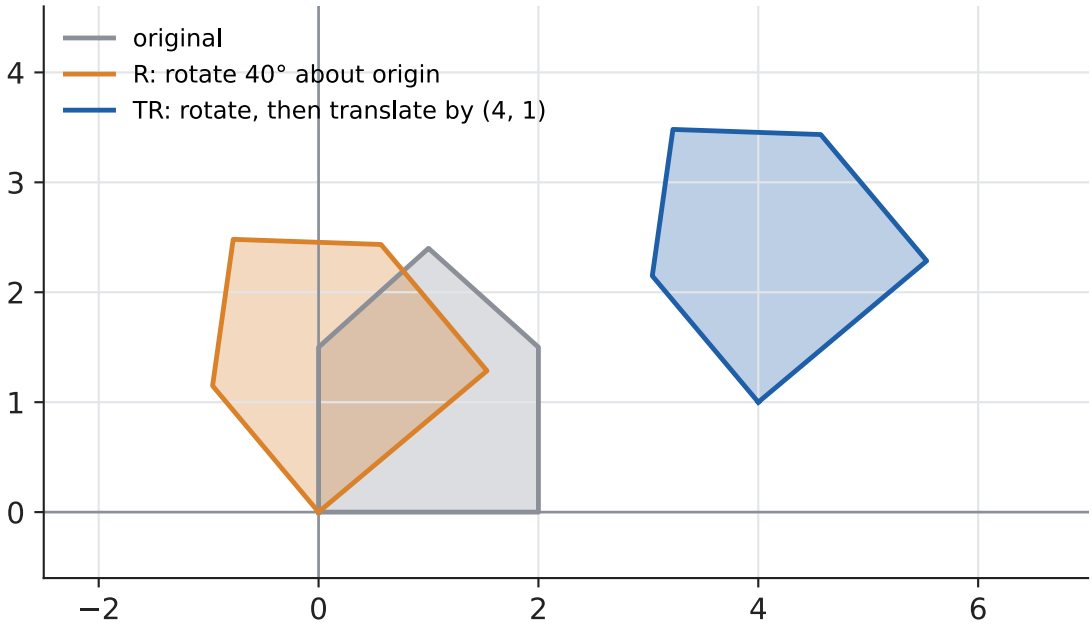


Figure 75. A house rotated 40° about the origin (orange), then translated by $(4, 1)$ (blue), using a single 3×3 homogeneous matrix TR.

27.11.3. Worked example: rotate about a point

Rotate the points $(2, 1)$, $(3, 1)$, $(3, 2)$ by $90°$ about the point $(2, 1)$. Plan (read right to left): translate by $(-2, -1)$ so the pivot is at the origin, rotate, translate back. *Try it:* Rotate then translate the house ; Rotate about a point .

$$M = T(2, 1)\ R_{90°}\ T(-2, -1).$$

By hand for $(3, 2)$: subtract the pivot, $(1, 1)$; rotate by $90°$, $(p, q) \mapsto (-q, p)$, giving $(-1, 1)$; add the pivot back, $(1, 2)$. In code, all three points at once, stored as columns:

PYTHON

```python
import numpy as np

def translate(tx, ty):
    return np.array([[1, 0, tx],
                     [0, 1, ty],
                     [0, 0, 1.0]])

def rotate(deg):
    t = np.radians(deg)
    return np.array([[np.cos(t), -np.sin(t), 0],
                     [np.sin(t),  np.cos(t), 0],
                     [0,          0,         1.0]])

# Rotate 90 degrees about the point (2, 1), not about the origin:
# move (2, 1) to the origin, rotate, move back. Read right to left.
M = translate(2, 1) @ rotate(90) @ translate(-2, -1)

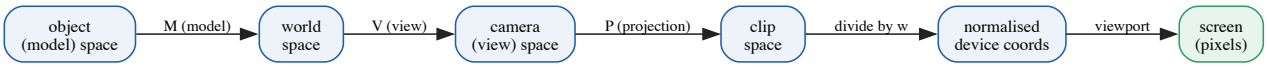
corners = np.array([[2, 3, 3],      # x-coordinates of three points
                    [1, 1, 2],      # y-coordinates
                    [1, 1, 1]])     # the extra 1s: "these are points"
print(np.round(M @ corners, 10))
```

```
[[2. 2. 1.]
 [1. 2. 2.]
 [1. 1. 1.]]
```

The pivot $(2, 1)$ stays fixed, $(3, 1)$ swings up to $(2, 2)$, and $(3, 2)$ goes to $(1, 2)$, matching the hand calculation. (The `np.round` hides rounding noise like `6e-17` from $\cos 90°$ in floating point.)

Computer-science connection: the graphics pipeline

Every vertex in a 3D game passes through a chain of 4×4 homogeneous matrices:



- The **model** matrix M places an object in the world (scale, rotate, translate).
- The **view** matrix V moves the world so the camera sits at the origin looking down an axis. It's the **inverse** of the camera's own placement matrix: moving the camera left is the same as moving the world right.
- The **projection** matrix P sets up perspective. It copies depth into the fourth (w) coordinate, and the GPU then divides x and y by w: distant things (large w) shrink. That division is the one non-linear step, and homogeneous coordinates make it a single cheap operation at the end.

The engine multiplies PVM once per object per frame (associativity at work), and the GPU's vertex shader applies that single matrix to every vertex: `gl_Position = mvp * vec4(position, 1.0);`. Normal vectors, which are directions, use 0 as the fourth coordinate (and, if the model matrix contains non-uniform scaling, need the inverse-transpose matrix to stay perpendicular to their surfaces).

Image transforms work in reverse. To rotate a photo, you don't push each source pixel forward (that leaves gaps); you loop over every **output** pixel, apply the **inverse** matrix to find where it came from in the source, and interpolate between the nearest source pixels. Image libraries' "affine warp" functions take exactly a 2×3 or 3×3 matrix.



Why "homogeneous"?

Homogeneous coordinates were introduced by August Möbius in 1827 for **projective geometry**, the mathematics of perspective drawing that Renaissance painters had worked out by eye. In projective geometry, parallel railway lines really do meet at the horizon: the meeting point is a "point at infinity" with final coordinate 0. The name refers to the equations becoming **homogeneous** (every term of the same degree). It took 140 years for the idea to find its killer application, when Lawrence Roberts used it for computer graphics at MIT in the 1960s.

27.12. Neural network layers: matrix multiply plus a bend

27.12.1. The problem

A neural network takes a vector in (an image's pixels, a sentence's embedding) and produces a vector out (class probabilities, a next-word distribution). In between are **layers**. What does a layer actually compute?

27.12.2. A layer is a matrix, then a nonlinearity

A **dense** (fully connected) layer from n inputs to m outputs computes

$$\mathbf{h} = \sigma(W\mathbf{x} + \mathbf{b}),$$

where W is an $m \times n$ **weight matrix**, $\mathbf{b}$ an m-vector of **biases**, and σ a simple function applied to each entry separately, such as the **ReLU** $\sigma(z) = \max(0, z)$. Read it with this chapter's eyes:

- $W\mathbf{x}$ is a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$. By the row view, output i is the dot product of row i of W with $\mathbf{x}$: each output "neuron" measures how much the input aligns with a learned direction, like the similarity scores in Chapter 26.
- $+\,\mathbf{b}$ is a translation (so the whole thing is **affine**, exactly what homogeneous coordinates handle).
- σ bends space. Without it, stacking layers would be pointless: $W_2(W_1\mathbf{x}) = (W_2W_1)\mathbf{x}$, a single matrix, because a composition of linear maps is linear. The nonlinearity is what lets deep networks represent curved decision boundaries.

Training adjusts the entries of W and $\mathbf{b}$ using gradients (Chapter 20). But **running** a network, called **inference**, is almost entirely matrix multiplication.

PYTHON

```python
import numpy as np

def relu(z):
    return np.maximum(0, z)

rng = np.random.default_rng(0)
W1, b1 = rng.standard_normal((4, 3)), np.zeros(4)   # layer 1: R^3 -> R^4
W2, b2 = rng.standard_normal((2, 4)), np.zeros(2)   # layer 2: R^4 -> R^2

x = np.array([1.0, 0.5, -2.0])       # one input with 3 features
h = relu(W1 @ x + b1)                # hidden layer: 4 numbers
y = W2 @ h + b2                      # output: 2 numbers
print(h.shape, y.shape)

X = rng.standard_normal((32, 3))     # a batch of 32 inputs, one per row
H = relu(X @ W1.T + b1)              # all 32 at once: one matrix multiply
print(H.shape)
```

```
(4,) (2,)
(32, 4)
```

Try it: Two linear layers collapse .

The last two lines show the trick that makes GPUs so effective: instead of 32 matrix–vector products, stack the inputs as rows of a matrix X and do one matrix–matrix product. Note the transpose: with inputs as **rows**, XW_1^T has row i equal to $(W_1\mathbf{x}_i)^\mathsf{T}$. Mismatched shapes in exactly this spot are the most common bug in hand-written ML code; when NumPy complains, write the shapes of every factor down and check the inner dimensions.

Computer-science connection: counting the cost of a model

A layer mapping 768 dimensions to 3072 has a 3072×768 weight matrix: about 2.4 million numbers, and 2.4 million multiply-adds per input vector. A transformer language model is stacks of such layers. Rule of thumb: generating one token with a model of P parameters costs about $2P$ floating-point operations, because each weight takes part in one multiply and one add. Knowing that "inference is mostly matrix–vector products, training mostly matrix–matrix products" explains most of the hardware story: memory bandwidth limits the former, raw arithmetic the latter.



Neurons as directions

A 2D dataset of two interleaved classes (for example two spirals). The learner picks a 2×2 matrix and bias for one layer and watches the data transform; then toggles ReLU on and off and sees how the bend folds the plane. A second layer follows. The app shows that without ReLU, two layers collapse to one matrix (it displays $W_2 W_1$), while with ReLU the classes can be pulled apart.

Try it: Neurons as directions: ReLU fold

27.13. Exercises: transpose, homogeneous coordinates, layers

1. **(Recognition)** If A is 3×5, what shape is $A^{\mathsf{T}}A$? AA^{T}? Are they symmetric?
2. **(Application)** Write the 3×3 homogeneous matrix that scales by 2 about the origin and then translates by $(1, -3)$, and apply it to the point $(4, 5)$ and to the direction $(4, 5)$.
3. **(Application)** Verify that $R_{30°}^{\mathsf{T}} R_{30°} = I$ by multiplying out with $c = \cos 30°$, $s = \sin 30°$.
4. **(Problem solving)** A network layer computes $W_2(W_1 \mathbf{x})$ with no nonlinearity, where W_1 is 2×100 and W_2 is 100×2. What is the largest possible rank of the combined matrix, and what does that say about this "network"?

▼ **Answers**

1. $A^{\mathsf{T}}A$ is $(5 \times 3)(3 \times 5) = 5 \times 5$; AA^{T} is 3×3. Both are symmetric: $(A^{\mathsf{T}}A)^{\mathsf{T}} = A^{\mathsf{T}}(A^{\mathsf{T}})^{\mathsf{T}} = A^{\mathsf{T}}A$.
2. $M = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & -3 \\ 0 & 0 & 1 \end{bmatrix}$. Point: $M(4, 5, 1) = (8 + 1, 10 - 3, 1) = (9, 7, 1)$, i.e. $(9, 7)$. Direction: $M(4, 5, 0) = (8, 10, 0)$: scaled but not translated.
3. $R^{\mathsf{T}}R = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} c^2 + s^2 & -cs + sc \\ -sc + cs & s^2 + c^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, using $\sin^2 + \cos^2 = 1$.

4. W_2W_1 is 100×100, but every input first passes through the 2-dimensional bottleneck $W_1\mathbf{x}$, so the rank is at most 2. The 100-dimensional output always lies in a plane. Without a nonlinearity, depth adds nothing: the two layers are one rank-≤2 linear map.

27.14. Summary

- A matrix is a **linear transformation**: it fixes the origin and keeps grid lines straight, parallel and evenly spaced. It can rotate, scale, shear, reflect and project. *Try it:* Scale then translate (exercise) .
- **Columns are where the basis vectors land.** $A\mathbf{x}$ is a combination of the columns (geometric view) or a list of row-dot-$\mathbf{x}$ values (computational view).
- **Matrix multiplication is composition**: AB means "B then A". It's associative but not commutative, and costs $O(n^3)$ naively.
- The **rotation matrix** $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ comes straight from the unit circle; composing rotations yields the angle-addition formulas.
- The **determinant** is the signed area/volume scale factor; $ad - bc$ for 2×2. Zero means **singular**: space is squashed, information lost, no inverse.
- The **inverse** undoes a transformation; for 2×2, swap the diagonal, negate the off-diagonal, divide by $\det$. To solve $A\mathbf{x} = \mathbf{b}$ in code, use `solve`, not `inv`.
- **Rank** counts the dimensions that survive. The **transpose** turns dot products into matrix products and inverts rotations.
- **Homogeneous coordinates** make translation (and perspective) a matrix, so graphics pipelines are chains of 4×4 matrices. **Neural network layers** are matrix multiplies with a nonlinearity between them.

27.15. Interactive tools

Browser tools on the companion website that go with this chapter. Each example link in the chapter opens one of them with that example already set up.

- Matrix transformations
- Matrix calculator

27.16. Mastery check

1. Write down the matrix that sends $\mathbf{e}_1 \mapsto (0, 2)$ and $\mathbf{e}_2 \mapsto (-1, 0)$. Describe the transformation in words, and give its determinant.
2. Compute $\begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ by both the row view and the column view.
3. Let $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Compute AB and BA, and describe geometrically what multiplying by B on each side does.
4. Find A^{-1} for $A = \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}$ and use it to solve $2x + 3y = 7$, $x + 2y = 4$.
5. Explain, using the picture of a transformation, why a matrix with determinant 0 cannot have an inverse.
6. Derive the matrix for an anticlockwise rotation by θ from scratch, and use it to rotate $(1, 1)$ by $45°$.
7. Give the 3×3 homogeneous matrix that translates by $(5, -2)$, and explain why no 2×2 matrix can do the same job.
8. A layer computes $\mathbf{ReLU}(W\mathbf{x} + \mathbf{b})$ with $\mathbf{x} \in \mathbb{R}^{512}$ and output in $\mathbb{R}^{128}$. What are the shapes of W and $\mathbf{b}$, and how many multiplications does $W\mathbf{x}$ need?

Decision rule: All correct without hesitation → move on to Chapter 28. Stuck on 1–2 or 6 → reread Section 27.2 and Section 27.3. Stuck on 3 → reread Section 27.5. Stuck on 4–5 → reread Section 27.6 and Section 27.7. Stuck on 7–8 → reread Section 27.11 and Section 27.12.

▼ **Mastery check answers**

1. $\begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix}$. $\mathbf{e}_1$ goes to the positive y-axis and $\mathbf{e}_2$ to the negative x-axis: a $90°$ anticlockwise rotation combined with stretching the original x-direction by 2 (precisely: scale x by 2, then rotate $90°$; check: $R_{90} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix}$). $\det = 0 \cdot 0 - (-1)(2) = 2$: areas double, orientation kept.
2. Rows: $(1 \cdot 3 + 2 \cdot 4,\ 0 \cdot 3 + (-1) \cdot 4) = (11, -4)$. Columns: $3(1, 0) + 4(2, -1) = (3 + 8, 0 - 4) = (11, -4)$.
3. $AB = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}$ (the columns of A swapped), $BA = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix}$ (the rows of A swapped). B swaps coordinates. In AB, B acts first on the input, swapping which basis vector goes where; in BA, it acts last, swapping the output's coordinates. Different, so $AB \neq BA$.

4. $\det A = 4 - 3 = 1$, $A^{-1} = \begin{bmatrix} 2 & -3 \\ -1 & 2 \end{bmatrix}$. $\mathbf{x} = A^{-1}(7, 4) = (14 - 12,\ -7 + 8) = (2, 1)$. Check: $4 + 3 = 7$, $2 + 2 = 4$. ✓

5. Determinant 0 means areas become 0: the plane is squashed onto a line (or a point). Then many different inputs land on the same output (whole lines of them collapse onto single points), so given an output you can't tell which input it came from. An inverse would have to send one point to many, which no function can do.

6. $\mathbf{e_1} = (1, 0)$ rotates to $(\cos\theta, \sin\theta)$ by definition of cos and sin on the unit circle; $\mathbf{e_2}$ at $90°$ rotates to angle $90° + \theta$, i.e. $(-\sin\theta, \cos\theta)$. These are the columns: $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$. At $45°$, $c = s = \frac{\sqrt{2}}{2}$: $(c - s,\ s + c) = (0, \sqrt{2})$. Length $\sqrt{2}$ preserved; the vector now points straight up. ✓

7. $\begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$. Any 2×2 matrix sends $\mathbf{0}$ to $\mathbf{0}$, but a translation moves the origin to $(5, -2)$, so translation is not linear and no 2×2 matrix represents it.

8. W is 128×512, $\mathbf{b}$ has 128 entries. $W\mathbf{x}$ needs $128 \times 512 = 65\,536$ multiplications.

Chapter 28. Eigenvectors, Bases, and Applications

A matrix moves almost every vector off its original line: it turns it as well as stretching it. But most matrices have a few special directions that they only stretch. Along those directions, a complicated transformation acts like plain multiplication by a number. Those directions are eigenvectors, and the numbers are eigenvalues. Find them, and hard questions become easy: what happens if you apply a matrix a thousand times? Where does a random process settle down? Which web page matters most? Which direction in a cloud of data carries the most information? This chapter builds eigenvectors from that one geometric question, computes them by hand for 2×2 matrices, and then follows them into Fibonacci numbers, Markov chains, PageRank, principal component analysis, and the singular value decomposition that compresses images and finds hidden topics in text.

Where this chapter sits

You need: matrices as transformations, matrix multiplication, determinants and inverses (Chapter 27); vectors, dot products and projections (Chapter 26); solving quadratic equations (Volume 1, Chapter 8); and for two applications, recurrences (Volume 1, Chapter 14) and directed graphs (Volume 3, Chapter 25). Complex numbers (Volume 1, Chapter 15) appear briefly. No calculus.

This chapter unlocks: the statistics of correlated data and regression (Volume 3, Chapter 31), stability of numerical methods (Chapter 32), the second-derivative test in several variables (Chapter 20), and the machine-learning and information-retrieval material in Volume 3, Chapter 33 .

Why this chapter matters. Many real questions are about what happens when the same step is repeated: next year's population from this year's, tomorrow's weather from today's, the next vibration of a bridge from the last, the next click of a web surfer. Eigenvectors answer "where does this end up, and how fast?" without simulating every step. The same idea, aimed at a cloud of data instead of a process, finds its most informative directions, which is how faces, genomes and documents get compressed into a few meaningful numbers.

28.1. Directions that only stretch

28.1.1. The problem

Take the matrix

$$A = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}.$$

Apply it to $(1, 0)$ and you get $(4, 2)$: the vector has been stretched **and** turned. Apply it to $(0, 1)$: $(1, 3)$, also turned. But apply it to $(1, 1)$:

$$\begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix} = 5 \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

No turning at all. Along the line through $(1, 1)$, this matrix does nothing except multiply by 5. Try $(1, -2)$: $A(1, -2) = (4 - 2, 2 - 6) = (2, -4) = 2(1, -2)$. Another unturned direction, stretched by 2.

Now here's why this matters. Suppose you want $A^{100}\mathbf{x}$, the result of applying A a hundred times. Multiplying out 100 matrices is tedious. But if $\mathbf{x} = (1, 1)$, the answer is instant: each application multiplies by 5, so $A^{100}(1, 1) = 5^{100}(1, 1)$. And since $(1, 1)$ and $(1, -2)$ point in different directions, **every** vector is a combination of them, and we can handle every vector this way. A complicated matrix has become two independent multiplications by numbers.

The questions for this chapter: how do you find such directions systematically, what do they mean, and where does this trick pay off?

28.1.2. Intuition: watch which arrows stay on their lines

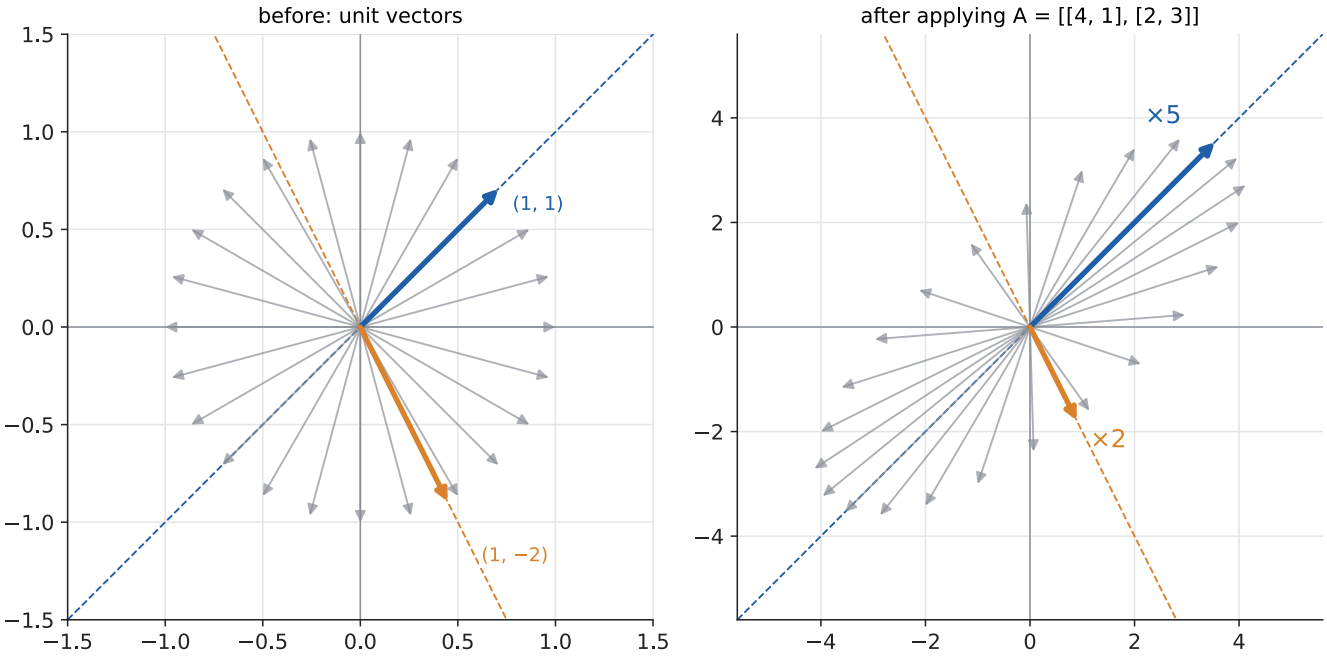


Figure 76. Unit vectors in every direction (left) and their images under A (right). Almost every arrow is turned. Only the two eigen-directions, through $(1, 1)$ (blue) and $(1, -2)$ (orange), stay on their own lines: they are stretched by 5 and 2.

Picture the transformation acting on the whole plane. Most directions get dragged around. But a few lines through the origin are mapped **onto themselves**: vectors on those lines stay on them, just longer, shorter, or flipped. Those are the transformation's natural axes, its "grain". A shear, a stretch, a projection: each has its own.

Eigenvector and eigenvalue

A nonzero vector $\mathbf{v}$ is an **eigenvector** of a square matrix A if

$$A\mathbf{v} = \lambda\mathbf{v}$$

for some scalar λ (Greek "lambda"), called the **eigenvalue** belonging to $\mathbf{v}$. In words: A maps $\mathbf{v}$ to a multiple of itself. $\lambda > 1$ stretches, $0 < \lambda < 1$ shrinks, $\lambda < 0$ flips (and scales by $|\lambda|$), $\lambda = 0$ crushes $\mathbf{v}$ to zero. Any nonzero multiple of an eigenvector is also an eigenvector with the same eigenvalue, so really an eigenvector names a **direction** (a line through the origin). The zero vector is never counted as an eigenvector, since $A\mathbf{0} = \lambda\mathbf{0}$ for every λ and tells you nothing.

Why "eigen"?

Eigen is German for "own" or "characteristic", as in *mein eigenes Auto* ("my own car"). An eigenvector is the matrix's **own** vector, a direction that belongs to it. David Hilbert popularised the term around 1904 in his work on integral equations. English briefly tried "proper value" and "characteristic value" (the latter survives in "characteristic equation"), but the half-German hybrid won. Physicists use "eigenstate", "eigenfrequency" and "eigenmode"; quantum mechanics is, in a sense, applied eigenvector theory.

28.1.3. A gallery of eigen-directions

You can often spot eigenvectors from the geometry alone.

- **Scaling** $\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$: the axes. $\mathbf{e_1}$ with $\lambda = 3$, $\mathbf{e_2}$ with $\lambda = 2$. **A diagonal matrix's eigenvectors are the coordinate axes and its eigenvalues are the diagonal entries.**
- **Reflection** in the line $y = x$: vectors on that line are unchanged ($\lambda = 1$); vectors perpendicular to it, along $(1, -1)$, are flipped ($\lambda = -1$).
- **Projection** onto the x-axis: $\mathbf{e_1}$ is kept ($\lambda = 1$); $\mathbf{e_2}$ is crushed ($\lambda = 0$). An eigenvalue of 0 means the matrix is singular.
- **Rotation** by $90°$: **every** vector is turned, so there are no real eigenvectors at all. (The eigenvalues turn out to be the complex numbers $\pm i$; see the failure modes below.)
- **Shear** $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$: the x-axis slides along itself unchanged ($\lambda = 1$), and **nothing else** stays on its line. Only one eigen-direction.

28.1.4. Where you'd use this

- **Everyday: why things shake at particular speeds.** A washing machine judders at one spin speed and runs smoothly above it; a car's mirror buzzes at one engine speed; a wine glass rings at one note. Each structure has **natural modes** of vibration, which are the eigenvectors of its stiffness-

and-mass equations, and the eigenvalues give the natural frequencies. Push at one of those frequencies and the motion builds up (**resonance**).

- **Science and engineering:** London's Millennium Bridge swayed alarmingly on its opening day in 2000 because the footfall of crowds matched a sideways natural mode near 1 Hz; it was closed for nearly two years while dampers were fitted. Structural engineers compute the lowest eigenvalues of a building model to keep its natural frequencies away from earthquake and wind frequencies. In quantum mechanics the allowed energies of an atom are eigenvalues, and the spectral lines astronomers use to identify elements in distant stars are differences between them. A tumbling object (throw a phone or a tennis racket in the air) spins steadily only about the eigenvectors of its **inertia matrix**, its principal axes.
- **Computing:** stability analysis. A numerical simulation, a control system or a recurrent neural network repeatedly multiplies by some matrix; if any eigenvalue has size greater than 1, errors grow without limit. Checking the largest eigenvalue is how engineers tell whether an autopilot, a physics time-step or a training run will settle or blow up (Chapter 32).

28.2. Finding eigenvalues: the characteristic equation

28.2.1. Why it works

We want nonzero $\mathbf{v}$ with $A\mathbf{v} = \lambda\mathbf{v}$. Rewrite that with everything on one side. Since $\lambda\mathbf{v} = \lambda I\mathbf{v}$,

$$\begin{aligned} A\mathbf{v} - \lambda I\mathbf{v} &= \mathbf{0} \\ (A - \lambda I)\,\mathbf{v} &= \mathbf{0}. \qquad \text{(factor out } \mathbf{v}) \end{aligned}$$

Read it as a transformation: the matrix $A - \lambda I$ sends the **nonzero** vector $\mathbf{v}$ to zero. An invertible matrix can't do that (it only sends $\mathbf{0}$ to $\mathbf{0}$; see Chapter 27). So $A - \lambda I$ must be singular, which means its determinant is zero:

$$\boxed{\det(A - \lambda I) = 0}$$

This is the **characteristic equation**. It contains only λ, not $\mathbf{v}$, so you can solve it for the eigenvalues first and find the eigenvectors afterwards. For a 2×2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$,

$$\begin{aligned} \det \begin{bmatrix} a-\lambda & b \\ c & d-\lambda \end{bmatrix} &= (a-\lambda)(d-\lambda) - bc \\ &= \lambda^2 - (a+d)\lambda + (ad - bc) = 0. \end{aligned}$$

The coefficients are two old friends: $a + d$ is the **trace** (sum of the diagonal) and $ad - bc$ is the determinant. So for any 2×2 matrix,

$$\lambda^2 - (\operatorname{tr} A)\,\lambda + \det A = 0,$$

a quadratic, solved with the quadratic formula (Volume 1, Chapter 8). By the sum and product of roots, $\lambda_1 + \lambda_2 = \operatorname{tr} A$ and $\lambda_1\lambda_2 = \det A$, two handy checks. The second also makes geometric sense: if A stretches one direction by λ_1 and another by λ_2, areas scale by $\lambda_1\lambda_2$.

Characteristic polynomial, trace

- The **characteristic polynomial** of an $n \times n$ matrix is $\det(A - \lambda I)$, a polynomial of degree n in λ. Its roots are the eigenvalues; setting it to zero gives the **characteristic equation**.
- The **trace** $\operatorname{tr} A$ is the sum of the diagonal entries. It equals the sum of the eigenvalues; the determinant equals their product.
- The **eigenspace** of λ is the set of all solutions of $(A - \lambda I)\mathbf{v} = \mathbf{0}$: all eigenvectors for λ together with $\mathbf{0}$.
- The set of all eigenvalues is the **spectrum** of A.

Example 35. Procedure — eigenvalues and eigenvectors of a 2×2 matrix

Input: $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

Steps:

1. Form the characteristic equation $\lambda^2 - (a+d)\lambda + (ad - bc) = 0$.
2. Solve it for λ_1, λ_2 (factor, or use the quadratic formula). Check: $\lambda_1 + \lambda_2 = a + d$ and $\lambda_1 \lambda_2 = ad - bc$.
3. For each eigenvalue λ, write down $A - \lambda I$. Its two rows are multiples of each other (it's singular), so they give a single equation in (x, y).
4. Solve that one equation for a nonzero vector. Shortcut: for a row (p, q) that isn't all zero, $(-q, p)$ (or $(q, -p)$) solves $px + qy = 0$.
5. Check $A\mathbf{v} = \lambda\mathbf{v}$.

Output: eigenvalues λ_1, λ_2 and eigenvectors $\mathbf{v}_1, \mathbf{v}_2$ (each determined up to a nonzero multiple).

Try it: Run the 2×2 procedure .

28.2.2. Worked example: the opening matrix

Find the eigenvalues and eigenvectors of $A = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$. *Try it:* Watch the eigenvectors stay put ; Step through this example .

Step 1: characteristic equation. Trace $= 4 + 3 = 7$, determinant $= 4 \cdot 3 - 1 \cdot 2 = 10$.

$$\begin{aligned}\det\begin{bmatrix}4-\lambda & 1\\ 2 & 3-\lambda\end{bmatrix} &= (4-\lambda)(3-\lambda) - 1\cdot 2\\ &= 12 - 7\lambda + \lambda^2 - 2 \qquad \text{(expand)}\\ &= \lambda^2 - 7\lambda + 10 = 0.\end{aligned}$$

Step 2: solve. We need two numbers with product 10 and sum 7: 5 and 2. So $(\lambda-5)(\lambda-2)=0$, giving $\lambda_1 = 5$ and $\lambda_2 = 2$. Checks: $5+2=7$ ✓, $5\times 2 = 10$ ✓.

Step 3–4: eigenvector for $\lambda_1 = 5$.

$$A - 5I = \begin{bmatrix}-1 & 1\\ 2 & -2\end{bmatrix}.$$

The second row is -2 times the first, as promised. The first row says $-x + y = 0$, i.e. $y = x$. Take $\mathbf{v}_1 = (1, 1)$.

Eigenvector for $\lambda_2 = 2$.

$$A - 2I = \begin{bmatrix}2 & 1\\ 2 & 1\end{bmatrix}.$$

Both rows say $2x + y = 0$, i.e. $y = -2x$. Take $\mathbf{v}_2 = (1, -2)$.

Step 5: check. $A(1,1) = (5,5) = 5(1,1)$ ✓ and $A(1,-2) = (2,-4) = 2(1,-2)$ ✓. These are exactly the directions highlighted in the diagram.

28.2.3. Alternative approaches, and choosing between them

- **By hand, 2×2:** the characteristic equation, as above. It's quick and shows the structure.
- **By hand, 3×3:** the characteristic polynomial is a cubic, which is unpleasant unless you can spot a root (try small integers, and remember the trace and determinant checks). Exam questions are designed so you can; real matrices aren't.
- **By computer, any size:** never through the characteristic polynomial. Finding polynomial roots is numerically ill-conditioned: tiny rounding errors in the coefficients move the roots a lot. Libraries use the **QR algorithm**, an iterative process of repeated orthogonal transformations, which is stable and $O(n^3)$. In NumPy: `np.linalg.eig` for general matrices, `np.linalg.eigh` for symmetric ones (faster, more accurate, and returns real, sorted, orthonormal results).
- **Only the largest eigenvector of a huge sparse matrix: power iteration** (Section 28.7), which needs nothing but matrix–vector products.



When eigenvectors go missing

- **Complex eigenvalues.** For the 90° rotation $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$, trace 0 and determinant 1 give $\lambda^2 + 1 = 0$, so $\lambda = \pm i$ (Volume 1, Chapter 15). No real direction is left unturned. Complex eigenvalues always signal rotation: their angle in the complex plane is the rotation angle, their size the scaling. In dynamical systems they mean oscillation.
- **Too few eigenvectors.** The shear $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ has characteristic equation $(1 - \lambda)^2 = 0$: a repeated eigenvalue 1. But $A - I = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ gives only $y = 0$: a single eigen-direction. Such a matrix is called **defective**, and it cannot be diagonalised (next section).
- **Repeated eigenvalues aren't always a problem.** $2I$ has eigenvalue 2 twice and **every** vector is an eigenvector.
- **NumPy normalises and may flip signs.** `np.linalg.eig` returns unit-length eigenvectors as **columns**, and the sign is arbitrary: $(-0.707, -0.707)$ is just as correct as $(0.707, 0.707)$. Never compare eigenvectors from two runs or two libraries without allowing for sign.

Try it: Rotation: complex eigenvalues ; 2I: every vector works ; Shear: a defective matrix .

Experiment: Eigenvectors using Python

Verify the worked example, then probe the failure modes.

PYTHON

```
import numpy as np

A = np.array([[4.0, 1.0],
              [2.0, 3.0]])
vals, vecs = np.linalg.eig(A)
print(vals)
print(vecs)                    # eigenvectors are the COLUMNS, scaled to
length 1

v = vecs[:, 0]
print(A @ v, vals[0] * v)      # A v equals lambda v
```

```
[5. 2.]
[[ 0.70710678 -0.4472136 ]
 [ 0.70710678  0.89442719]]
[3.53553391 3.53553391] [3.53553391 3.53553391]
```



The columns are $(1,1)/\sqrt{2}$ and $(-1,2)/\sqrt{5}$, which is $-(1,-2)/\sqrt{5}$: the same direction as our hand answer, normalised and flipped. Now try these and explain each result geometrically:

1. `np.linalg.eig(np.array([[0.0, -1.0], [1.0, 0.0]]))` (a rotation: expect complex output).
2. `np.linalg.eig(np.array([[1.0, 1.0], [0.0, 1.0]]))` (a shear: look at the two eigenvector columns; are they really different?).
3. A random symmetric matrix `M = R + R.T` for a random `R`: compute `vecs.T @ vecs` and see what you get (preview of Section 28.9).
4. Pick any 2×2 matrix and any starting vector, apply the matrix 30 times (normalising each time), and compare the result with the eigenvector of the largest eigenvalue.

Try it: Verify it step by step · Run the four experiments

Try it: Run this in your browser .

US curriculum

Eigenvalues and eigenvectors are **not** part of the standard US high-school curriculum. They're a core topic of a first college **Linear Algebra** course (typically first or second year for STEM majors) and reappear in **Differential Equations**. Some advanced high schools offer post-AP linear algebra courses that include them.

28.3. Exercises: finding eigenvectors

1. **(Recognition)** Without calculating, give the eigenvalues and eigenvectors of $\begin{bmatrix} -1 & 0 \\ 0 & 7 \end{bmatrix}$ and of the reflection in the x-axis. *Try it:* Check exercise [1 2; 3 2) ; Check exercise [2 1; 1 2) .
2. **(Application)** Find the eigenvalues and eigenvectors of $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.
3. **(Application)** Find the eigenvalues of $\begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}$ and one eigenvector for each.
4. **(Problem solving)** A 2×2 matrix has trace 6 and determinant 9. What can you say about its eigenvalues? Give two different matrices with these properties, one with a single eigen-direction and one where every vector is an eigenvector.

▼ **Answers**

1. Diagonal: eigenvalues -1 (along $\mathbf{e}_1$) and 7 (along $\mathbf{e}_2$). Reflection in the x-axis: $\mathbf{e}_1$ with $\lambda = 1$, $\mathbf{e}_2$ with $\lambda = -1$.
2. Trace 4, determinant 3: $\lambda^2 - 4\lambda + 3 = (\lambda - 3)(\lambda - 1) = 0$, so $\lambda = 3, 1$. For 3: $A - 3I = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$, $y = x$, $\mathbf{v} = (1, 1)$. For 1: $A - I = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, $y = -x$, $\mathbf{v} = (1, -1)$. (Note: perpendicular.)
3. Trace 3, determinant $2 - 6 = -4$: $\lambda^2 - 3\lambda - 4 = (\lambda - 4)(\lambda + 1) = 0$. For 4: $A - 4I = \begin{bmatrix} -3 & 2 \\ 3 & -2 \end{bmatrix}$, $-3x + 2y = 0$, $\mathbf{v} = (2, 3)$. Check: $A(2, 3) = (2 + 6, 6 + 6) = (8, 12) = 4(2, 3)$ ✓. For -1: $A + I = \begin{bmatrix} 2 & 2 \\ 3 & 3 \end{bmatrix}$, $x + y = 0$, $\mathbf{v} = (1, -1)$. Check: $A(1, -1) = (-1, 1)$ ✓.
4. $\lambda^2 - 6\lambda + 9 = (\lambda - 3)^2$: a repeated eigenvalue 3. $3I$ has every vector as an eigenvector. $\begin{bmatrix} 3 & 1 \\ 0 & 3 \end{bmatrix}$ has trace 6, determinant 9, and only the eigen-direction $\mathbf{e}_1$ (since $A - 3I = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ forces $y = 0$).

28.4. Change of basis and diagonalisation

28.4.1. The problem

We found that $A = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$ simply multiplies by 5 along $(1, 1)$ and by 2 along $(1, -2)$. That suggests a strategy: **describe vectors in terms of those two directions instead of $\mathbf{e}_1, \mathbf{e}_2$.** In those coordinates, A should look like the diagonal matrix $\begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}$. How do we switch coordinates, and what does it buy us?

28.4.2. Change of basis

Any two independent vectors form a basis of the plane (Chapter 26). Put the new basis vectors as the columns of a matrix P:

$$P = \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix}.$$

If a vector has **coordinates** $\mathbf{c} = (c_1, c_2)$ in the new basis, meaning it equals $c_1(1,1) + c_2(1,-2)$, then in standard coordinates it's $\mathbf{x} = P\mathbf{c}$ (that's the column view of matrix–vector multiplication). Going the other way, $\mathbf{c} = P^{-1}\mathbf{x}$. So:

- P translates **from** new coordinates **to** standard coordinates.
- P^{-1} translates **from** standard coordinates **to** new coordinates.

Using the 2×2 inverse formula, $\det P = -2 - 1 = -3$ and

$$P^{-1} = \frac{1}{-3}\begin{bmatrix} -2 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & -1/3 \end{bmatrix}.$$

Example. The vector $\mathbf{x} = (3, 0)$ has new coordinates $P^{-1}(3,0) = (2,1)$. Check: $2(1,1) + 1(1,-2) = (3,0)$ ✓.

28.4.3. Diagonalisation: three moves

Now compute $A\mathbf{x}$ by a detour through eigen-coordinates:

1. Translate into eigen-coordinates: $\mathbf{c} = P^{-1}\mathbf{x}$.
2. In those coordinates A just scales each coordinate by its eigenvalue: multiply by $D = \begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}$.
3. Translate back: multiply by P.

So $A\mathbf{x} = PDP^{-1}\mathbf{x}$ for every $\mathbf{x}$, which means

$$\boxed{A = PDP^{-1}}$$

where the columns of P are eigenvectors and D is diagonal with the matching eigenvalues. Reading right to left: **change to the matrix's own coordinates, stretch along the axes, change back.** A complicated-looking transformation is a diagonal one viewed in a tilted coordinate system.

Check with $\mathbf{x} = (3, 0)$. Coordinates $(2, 1)$; scale to $(10, 2)$; back to standard: $10(1,1) + 2(1,-2) = (12, 6)$. Directly: $A(3,0) = (12, 6)$ ✓.

Diagonalisable, similar, change of basis

- A square matrix is **diagonalisable** if $A = PDP^{-1}$ with D diagonal. This works exactly when A has n linearly independent eigenvectors (the columns of P); the eigenvalues fill the diagonal of D in the same order.
- A **change-of-basis matrix** P has the new basis vectors as columns; $P\mathbf{c}$ converts new coordinates to standard ones, $P^{-1}\mathbf{x}$ converts back.
- Matrices A and $B = P^{-1}AP$ are **similar**: the same transformation described in different coordinates. Similar matrices share eigenvalues, trace and determinant.
- n **distinct** eigenvalues guarantee diagonalisability. Repeated eigenvalues may or may not allow it ($2I$ yes, the shear no).

28.4.4. Why powers become easy

Here's the payoff. Apply A twice:

$$A^2 = (PDP^{-1})(PDP^{-1}) = PD(P^{-1}P)DP^{-1} = PD\,I\,DP^{-1} = PD^2P^{-1}.$$

The inner $P^{-1}P$ cancels: there's no point changing back to standard coordinates between applications. The same happens for any power:

$$A^n = PD^nP^{-1}, \qquad D^n = \begin{bmatrix} 5^n & 0 \\ 0 & 2^n \end{bmatrix}.$$

Powers of a diagonal matrix are trivial: raise each diagonal entry to the power. So the cost of A^n no longer depends on n (beyond computing two numerical powers).

Example 36. Procedure — computing A^n by diagonalisation

Input: a diagonalisable $n \times n$ matrix A and a power k.

Steps:

1. Find the eigenvalues $\lambda_1, \ldots, \lambda_n$ and eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$.
2. Form $P = [\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_n]$ and $D = \mathbf{diag}(\lambda_1, \ldots, \lambda_n)$. Compute P^{-1}.
3. Form $D^k = \mathbf{diag}(\lambda_1^k, \ldots, \lambda_n^k)$.
4. Multiply $A^k = PD^kP^{-1}$.

Output: A^k, often as a closed-form formula in k. If you only need $A^k\mathbf{x}$: write $\mathbf{x} = \sum c_i\mathbf{v}_i$ (i.e. $\mathbf{c} = P^{-1}\mathbf{x}$), and then $A^k\mathbf{x} = \sum c_i\lambda_i^k\mathbf{v}_i$.

28.4.5. Worked example: a formula for A^n

$$PD^n = \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 5^n & 0 \\ 0 & 2^n \end{bmatrix} = \begin{bmatrix} 5^n & 2^n \\ 5^n & -2 \cdot 2^n \end{bmatrix} \qquad \text{(scale the columns)}$$

$$\begin{aligned} A^n = PD^nP^{-1} &= \begin{bmatrix} 5^n & 2^n \\ 5^n & -2 \cdot 2^n \end{bmatrix} \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & -1/3 \end{bmatrix} \\ &= \frac{1}{3} \begin{bmatrix} 2 \cdot 5^n + 2^n & 5^n - 2^n \\ 2 \cdot 5^n - 2 \cdot 2^n & 5^n + 2 \cdot 2^n \end{bmatrix}. \end{aligned}$$

Try it: Compute A³ directly ; Aⁿ by diagonalisation, live .

Check at $n = 1$: $\frac{1}{3}(12, 3; 6, 9) = (4, 1; 2, 3) = A$ ✓. **At $n = 3$:** top-left $(250 + 8)/3 = 86$, top-right $(125 - 8)/3 = 39$, bottom-left $(250 - 16)/3 = 78$, bottom-right $(125 + 16)/3 = 47$. Multiplying out $A \cdot A \cdot A$ directly gives $\begin{bmatrix} 86 & 39 \\ 78 & 47 \end{bmatrix}$ ✓.

The formula also tells you the **long-run behaviour** at a glance. For large n, 5^n dwarfs 2^n, so $A^n \approx \frac{5^n}{3} \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix}$: every vector gets pulled towards the $(1, 1)$ direction and grows like 5^n. **The largest eigenvalue governs the long run.** That single observation drives the next three sections.

28.4.6. Where you'd use this: a population model

Ecologists track animal populations by age class with a **Leslie matrix**. Suppose each adult produces on average 3 juveniles a year, half the juveniles survive to become adults, and half the adults survive each year. With $\mathbf{x}_n = (j_n, a_n)$ (juveniles, adults) in year n, *Try it:* Run the Leslie model ; Leslie matrix eigenvalues .

$$\mathbf{x}_{n+1} = \underbrace{\begin{bmatrix} 0 & 3 \\ 0.5 & 0.5 \end{bmatrix}}_{L} \mathbf{x}_n, \qquad \lambda^2 - 0.5\lambda - 1.5 = 0 \;\Rightarrow\; \lambda = \frac{0.5 \pm \sqrt{0.25 + 6}}{2} = 1.5 \text{ or } -1.$$

(Trace 0.5, determinant $0 - 1.5 = -1.5$.) For $\lambda = 1.5$, $L - 1.5I = \begin{bmatrix} -1.5 & 3 \\ 0.5 & -1 \end{bmatrix}$ gives $j = 2a$, eigenvector $(2, 1)$. So in the long run the population grows by 50% a year with two juveniles for every adult, whatever it started as. The eigenvalue -1 adds a boom-and-bust wobble that alternates sign each year, but relative to the 1.5^n growth it shrinks by a factor $1/1.5 = 2/3$ a year. Start with 10 adults and no juveniles: year by year the totals are 35, 32.5, 68.75, 83.1, 144.7, 197.0, and the juvenile-to-adult ratio goes $6, 0.86, 3.2, 1.42, 2.48, 1.72, \ldots$, closing in on 2. Fisheries managers use exactly this calculation, with more age classes, to set catch limits: harvest so that the dominant eigenvalue stays at about 1.

Beyond ecology, the same "largest eigenvalue governs the long run" reasoning gives compound growth of interacting quantities in economics, the spread rate of an epidemic across age groups (the famous R_0 is the dominant eigenvalue of a "next-generation matrix"), and the asymptotic running time of algorithms defined by recurrences, as the next section shows.



Diagonalisation fails for defective matrices

If there aren't n independent eigenvectors (the shear, or anything with a complex eigenvalue if you insist on real numbers), P isn't invertible and $A = PDP^{-1}$ is impossible. The general fallback is the **Jordan form** (nearly diagonal, with 1s just above the diagonal), which is theoretically important but numerically fragile; in practice, people use the **Schur decomposition** or the SVD (Section 28.11) instead. And even when A is diagonalisable, if its eigenvectors are nearly parallel then P is nearly singular and computing with PDP^{-1} loses accuracy. Symmetric matrices never have either problem.

28.5. Fibonacci via matrix powers

28.5.1. The problem

The Fibonacci sequence (Volume 1, Chapter 14) is $F_0 = 0, F_1 = 1, F_{n+1} = F_n + F_{n-1}$: 0, 1, 1, 2, 3, 5, 8, 13, … Is there a formula for F_n that doesn't require computing all the earlier terms? And how fast does the sequence grow?

28.5.2. Turning a recurrence into a matrix

The recurrence needs the two most recent terms, so track them together as a **state vector** (F_{n+1}, F_n). One step forward:

$$\begin{bmatrix} F_{n+2} \\ F_{n+1} \end{bmatrix} = \begin{bmatrix} F_{n+1} + F_n \\ F_{n+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}}_{M} \begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix}.$$

Starting from $(F_1, F_0) = (1, 0)$, n steps give $(F_{n+1}, F_n) = M^n(1, 0)$. So the Fibonacci numbers are the entries of the powers of M. (In fact $M^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$.)

28.5.3. Eigenvalues of M

Trace 1, determinant -1, so the characteristic equation is

$$\lambda^2 - \lambda - 1 = 0 \quad \Longrightarrow \quad \lambda = \frac{1 \pm \sqrt{5}}{2}.$$

These are the golden ratio $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ and its partner $\psi = \frac{1-\sqrt{5}}{2} \approx -0.618$. Eigenvectors: for either root λ, the vector $(\lambda, 1)$ works, because

$$M \begin{bmatrix} \lambda \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda + 1 \\ \lambda \end{bmatrix} = \begin{bmatrix} \lambda^2 \\ \lambda \end{bmatrix} = \lambda \begin{bmatrix} \lambda \\ 1 \end{bmatrix}, \qquad \text{using } \lambda^2 = \lambda + 1.$$

28.5.4. Deriving Binet's formula

Write the starting vector as a combination of eigenvectors. Since $\varphi - \psi = \sqrt{5}$,

$$\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{5}}\left(\begin{bmatrix}\varphi\\1\end{bmatrix} - \begin{bmatrix}\psi\\1\end{bmatrix}\right).$$

(Check: top $(\varphi - \psi)/\sqrt{5} = 1$, bottom $(1-1)/\sqrt{5} = 0$ ✓.) Now apply M^n: each eigenvector just gets multiplied by its eigenvalue to the n:

$$\begin{bmatrix}F_{n+1}\\F_n\end{bmatrix} = M^n\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{5}}\left(\varphi^n\begin{bmatrix}\varphi\\1\end{bmatrix} - \psi^n\begin{bmatrix}\psi\\1\end{bmatrix}\right).$$

The bottom entry is

$$\boxed{F_n = \frac{\varphi^n - \psi^n}{\sqrt{5}}}$$

This is **Binet's formula**. A sequence of whole numbers is given by a formula full of square roots of 5, which all cancel. Since $|\psi| < 1$, the ψ^n term shrinks to nothing, so F_n is simply $\varphi^n/\sqrt{5}$ rounded to the nearest integer, and the ratio of consecutive terms tends to φ. The growth rate of the sequence **is** the dominant eigenvalue.

28.5.5. Implementation: exact Fibonacci in $O(\log n)$ steps

Binet's formula is elegant but floating-point arithmetic ruins it for large n. Matrix powers computed by **repeated squaring** stay in exact integers and need only about $\log_2 n$ matrix multiplications (the same trick as fast modular exponentiation in Volume 3, Chapter 23): *Try it:* Matrix powers versus Binet's formula .

```python
def mat_mult(X, Y):
    return [[X[0][0]*Y[0][0] + X[0][1]*Y[1][0], X[0][0]*Y[0][1] + X[0][1]*Y[1][1]],
            [X[1][0]*Y[0][0] + X[1][1]*Y[1][0], X[1][0]*Y[0][1] + X[1][1]*Y[1][1]]]

def mat_pow(M, n):
    """M**n by repeated squaring: O(log n) matrix multiplications."""
    result = [[1, 0], [0, 1]]              # identity
    while n > 0:
        if n % 2 == 1:                     # this binary digit of n is 1
            result = mat_mult(result, M)
        M = mat_mult(M, M)                 # square for the next digit
        n //= 2
    return result

def fib(n):
    return mat_pow([[1, 1], [1, 0]], n)[0][1]

print([fib(n) for n in range(1, 13)])
print(fib(100))
phi = (1 + 5 ** 0.5) / 2
print(round(phi ** 100 / 5 ** 0.5))     # Binet, dominant term only
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
354224848179261915075
354224848179263111168
```

The exact answer and the floating-point Binet answer agree in their first 14 digits and then diverge: a 64-bit float carries only about 16 significant digits (Chapter 32). **Choose the method by the job**: Binet for growth rates and estimates, exact matrix powers for exact values.

Computer-science connection: linear recurrences everywhere

Any **linear recurrence**, where the next value is a fixed linear combination of the previous k values, becomes a $k \times k$ matrix power in the same way. Its long-run growth rate is the largest eigenvalue in absolute value. This is how you analyse the running time of recursive algorithms whose recurrences look like Fibonacci (the naive recursive `fib` makes about φ^n calls), count strings that avoid a forbidden pattern (Volume 3, Chapter 24), and compute population models such as the **Leslie matrix** in ecology. **Linear-feedback shift registers**, used for pseudo-random numbers and CRC checksums, are matrix powers over the integers mod 2.

28.6. Markov chains and steady states

28.6.1. The problem

A crude weather model: if today is sunny, tomorrow is sunny with probability 0.9 and rainy with probability 0.1. If today is rainy, tomorrow is sunny or rainy with probability 0.5 each. It's raining today. What's the chance of sun in a week? In the long run, what fraction of days are sunny?

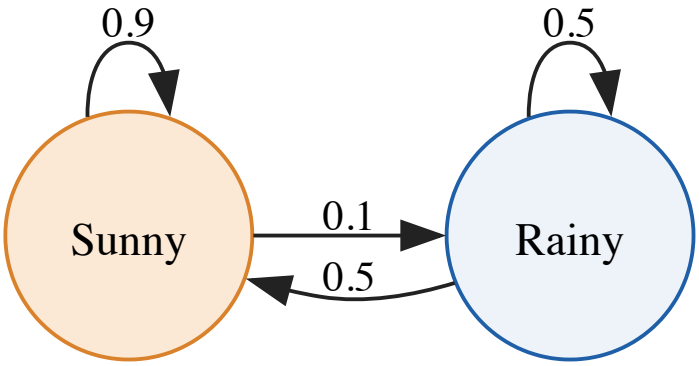


Figure 77. A two-state Markov chain. Each arrow is labelled with the probability of moving along it tomorrow; the probabilities leaving each state sum to 1.

28.6.2. Setting it up as a matrix

Let $\mathbf{x}_n = (s_n, r_n)$ be the probabilities of sun and rain on day n. Tomorrow's chance of sun comes from two routes, sunny-then-sunny and rainy-then-sunny:

$$\begin{aligned} s_{n+1} &= 0.9\,s_n + 0.5\,r_n \\ r_{n+1} &= 0.1\,s_n + 0.5\,r_n \end{aligned} \qquad \Longleftrightarrow \qquad \mathbf{x}_{n+1} = \underbrace{\begin{bmatrix} 0.9 & 0.5 \\ 0.1 & 0.5 \end{bmatrix}}_{P} \mathbf{x}_n .$$

Column j of P lists where you go from state j; each column sums to 1. After n days, $\mathbf{x}_n = P^n \mathbf{x}_0$: a matrix power again.



Markov chain, transition matrix, steady state

A **Markov chain** is a random process that moves between **states**, where the probability of the next state depends only on the current one (it has no memory of earlier history). Its **transition matrix** P has entry p_{ij} = probability of moving **to** state i **from** state j. With this convention (**column-stochastic**), columns sum to 1 and distributions update as $\mathbf{x} \mapsto P\mathbf{x}$. Many statistics texts use the transpose (**row-stochastic**, rows sum to 1, $\mathbf{x}^{\mathsf{T}} \mapsto \mathbf{x}^{\mathsf{T}} P$); check which a source uses. A **steady state** (or **stationary distribution**) is a probability vector with $P\mathbf{x} = \mathbf{x}$: an eigenvector with eigenvalue 1, scaled so its entries sum to 1.

28.6.3. The steady state is an eigenvector

If the distribution settles down to some $\mathbf{x}^*$, then applying P once more leaves it unchanged: $P\mathbf{x}^* = \mathbf{x}^*$. That's an eigenvector with eigenvalue 1. Solve $(P - I)\mathbf{x} = \mathbf{0}$:

$$P - I = \begin{bmatrix} -0.1 & 0.5 \\ 0.1 & -0.5 \end{bmatrix} \qquad \text{(rows are negatives of each other)}$$

$$-0.1\,s + 0.5\,r = 0 \quad \Longrightarrow \quad s = 5r,$$

$$s + r = 1 \quad \Longrightarrow \quad 6r = 1, \quad r = \tfrac{1}{6}, \quad s = \tfrac{5}{6}.$$

In the long run, five days in six are sunny, regardless of today's weather.

28.6.4. How fast it gets there: the second eigenvalue

Trace 1.4, determinant $0.45 - 0.05 = 0.4$, so $\lambda^2 - 1.4\lambda + 0.4 = (\lambda - 1)(\lambda - 0.4) = 0$. One eigenvalue is 1 (the steady state); the other is 0.4. Decompose the starting distribution into the steady state plus a component along the second eigenvector; each day that second component is multiplied by 0.4. So the distance from equilibrium shrinks by a factor of 0.4 per day. Starting from rain, the chance of sun on day n works out to *Try it:* Explore the weather chain .

$$s_n = \tfrac{5}{6} - \tfrac{5}{6}(0.4)^n,$$

which gives $s_1 = 0.5$, $s_2 = 0.7$, $s_3 = 0.78$, and by day 7, 0.8320: within 0.002 of $5/6$. The code confirms it:

PYTHON

```python
import numpy as np

P = np.array([[0.9, 0.5],     # column j = today's state j; rows = tomorrow
              [0.1, 0.5]])
x = np.array([0.0, 1.0])      # start: certainly rainy
for day in range(1, 7):
    x = P @ x
    print(day, x.round(4))

vals, vecs = np.linalg.eig(P)
print(vals)
k = np.argmax(np.isclose(vals, 1.0))
steady = vecs[:, k] / vecs[:, k].sum()   # rescale so entries sum to 1
print(steady)
```

```
1 [0.5 0.5]
2 [0.7 0.3]
3 [0.78 0.22]
4 [0.812 0.188]
5 [0.8248 0.1752]
6 [0.8299 0.1701]
[1.  0.4]
[0.83333333 0.16666667]
```

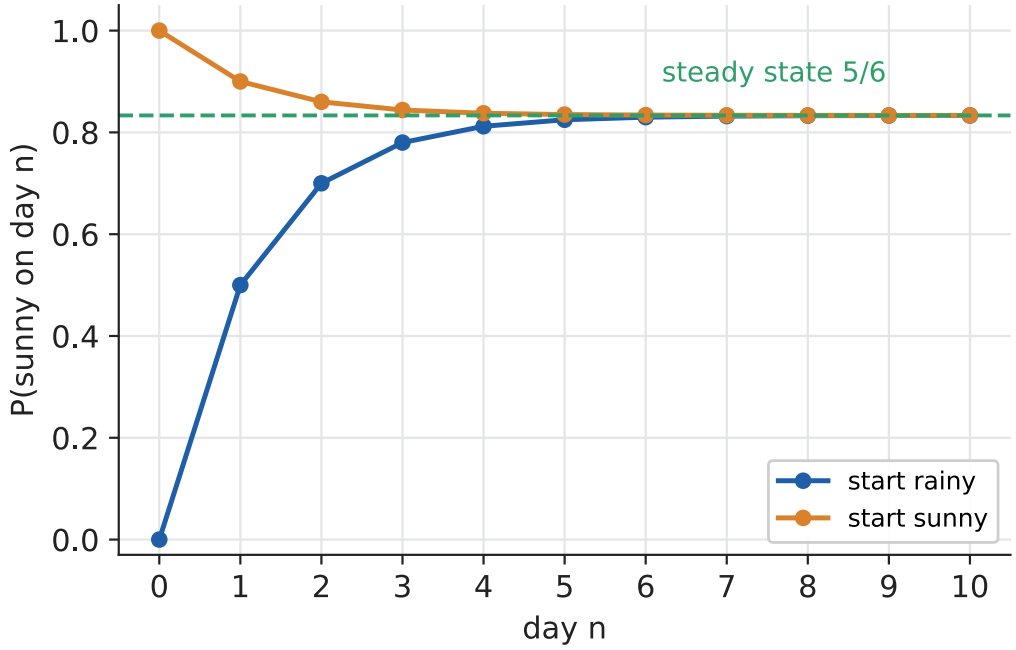


Figure 78. The chance of sun on day n, starting from a rainy day (blue) or a sunny day (orange). Both converge to the steady state $5/6$ (dashed), and the gap shrinks by the second eigenvalue's factor of 0.4 each day, so after a week the starting weather is all but forgotten.

Why is 1 always an eigenvalue of a transition matrix, and why are the others no bigger than 1 in size? Because the columns sum to 1, multiplying by P preserves total probability, so no component can grow forever. The **gap** between 1 and the next-largest eigenvalue size (here $1 - 0.4 = 0.6$) is the **spectral gap**, and it controls the convergence speed: a big gap means the chain forgets its starting point quickly.



When there's no single steady state

Two things can go wrong. A chain that **cycles** ($P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$: rain always follows sun and vice versa) has a second eigenvalue of -1, so the distribution oscillates forever and never settles, even though a stationary distribution $\left(\frac{1}{2}, \frac{1}{2}\right)$ exists. A chain that splits into **separate pieces** that can't reach each other has eigenvalue 1 more than once and several steady states, depending on where you start. A chain with neither problem (**irreducible** and **aperiodic**) always converges to a unique steady state from any starting distribution; that's the content of the Perron–Frobenius theorem. PageRank's "damping" trick exists precisely to force these conditions.

Try it: Watch a cycling chain .

28.6.5. Where you'd use this

- **Everyday: brand switching and board games.** Each month 20% of a phone network's customers leave for its rival, and 10% of the rival's customers come the other way. The transition matrix $\begin{bmatrix} 0.8 & 0.1 \\ 0.2 & 0.9 \end{bmatrix}$ has steady state from $0.2a = 0.1b$, so $b = 2a$: in the long run the first network keeps only $\frac{1}{3}$ of the market, wherever it starts. (Halving its churn from 20% to 10% would make the shares equal.) The same eigenvector calculation, with 40 states, shows which Monopoly squares players land on most often. *Try it:* Brand switching chain .
- **Science:** DNA sequence evolution models, chemical reaction networks, the queue at a call centre and credit-rating changes for bonds (rated AA this year, what's the chance of A next year?) are all Markov chains whose long-run behaviour is an eigenvector.
- **Computing:** your phone keyboard's next-word suggestions began as Markov chains on words; **Markov chain Monte Carlo** (MCMC) deliberately builds a chain whose steady state is a probability distribution you want to sample, and is the workhorse of Bayesian statistics (Volume 3, Chapter 29); and the random surfer of the next section is a Markov chain on web pages.

28.7. PageRank: the eigenvector that built Google

28.7.1. The problem

In 1998 the web had tens of millions of pages, and keyword search returned thousands of matches for any query. Which matches should come first? Larry Page and Sergey Brin's idea: a page is important if important pages link to it. That sounds circular, and it is. Eigenvectors are how you solve circular definitions.

28.7.2. The random surfer

Imagine a surfer who starts on a random page and, at every step, clicks a random link on the current page. Over a long time, what fraction of steps does the surfer spend on each page? Pages with many incoming links from frequently visited pages will be visited often. That long-run fraction is the page's **PageRank**, and it's exactly the steady state of a Markov chain whose states are web pages.

Here's a tiny web:

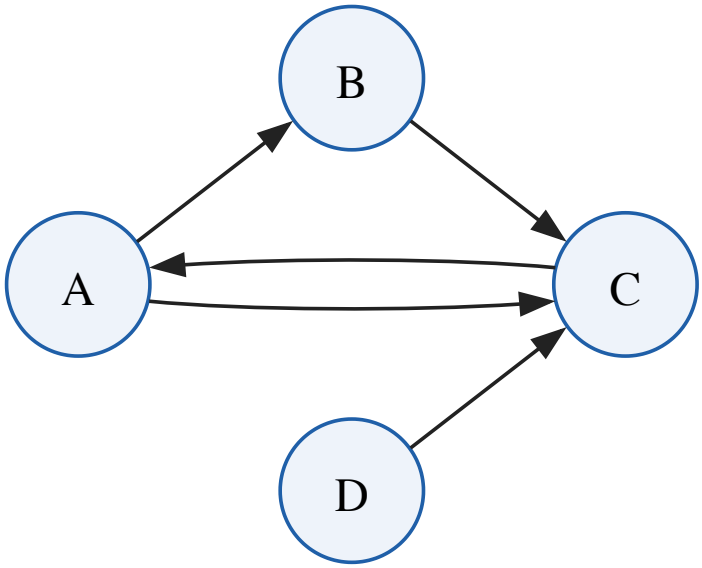


Figure 79. A four-page web. Page A links to B and C; B links to C; C links to A; D links to C, and nothing links to D.

The link matrix M has $m_{ij} = 1/(\text{number of out-links of } j)$ if page j links to page i, and 0 otherwise. Columns are pages being left; each column splits its page's "vote" equally among its links (Volume 3, Chapter 25 calls this a weighted adjacency matrix):

$$M = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \!\!\begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ \tfrac{1}{2} & 0 & 0 & 0 \\ \tfrac{1}{2} & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}.$$

28.7.3. Damping: the teleporting surfer

The pure random surfer gets stuck in pages with no out-links and in closed loops, and page D (no incoming links) would get rank zero, which breaks the uniqueness conditions above. The fix: with probability d (the **damping factor**, classically 0.85) the surfer follows a random link; with probability $1 - d$ they get bored and jump to a page chosen uniformly at random. With n pages, the **Google matrix** is *Try it:* Move the damping slider .

$$G = d\,M + \frac{1 - d}{n}\,J,$$

where J is the matrix of all 1s. Every entry of G is positive, so the Perron–Frobenius theorem guarantees a unique steady state with all-positive entries, and the chain converges to it. The PageRank vector $\mathbf{r}$ is the eigenvector of G with eigenvalue 1:

$$G\,\mathbf{r} = \mathbf{r}, \qquad \textstyle\sum_i r_i = 1.$$

28.7.4. Power iteration

For the real web, G has billions of rows. You can't run a textbook eigenvalue algorithm on it. But you don't need to: you know the eigenvalue you want (1), you know it's the largest, and the diagonalisation section showed that **repeatedly applying a matrix swings any vector towards its dominant eigenvector**. So just keep multiplying. *Try it:* Watch power iteration converge .

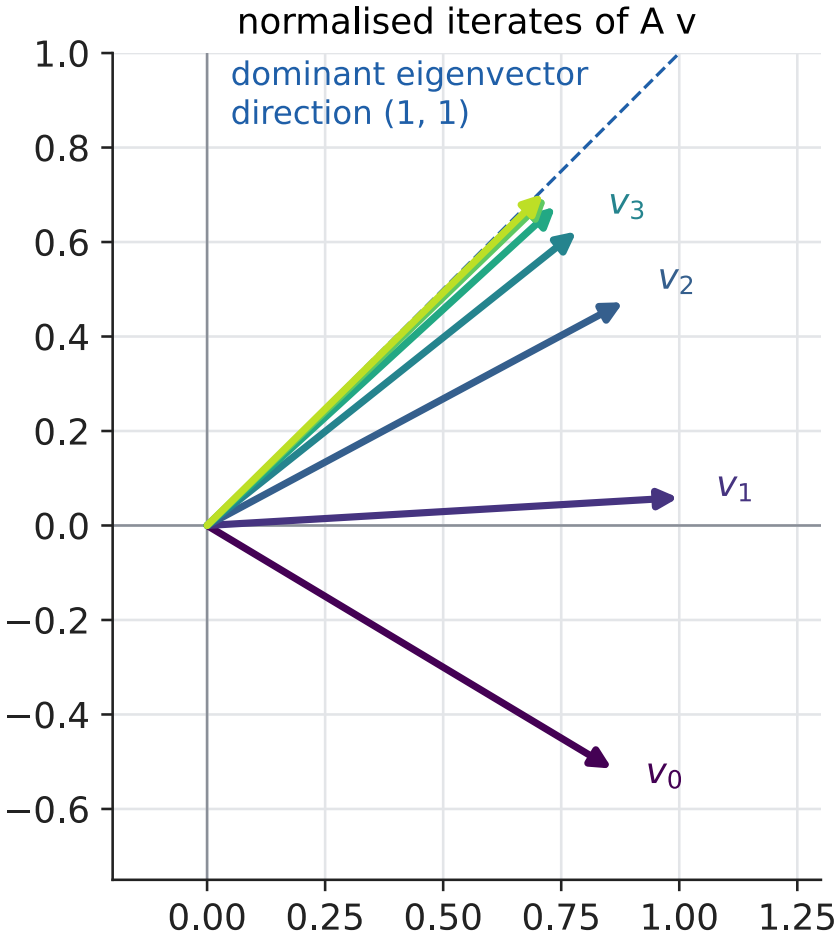


Figure 80. Power iteration: repeatedly applying $A = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$ and renormalising pulls a starting vector $\mathbf{v}_0$ onto the dominant eigenvector direction $(1, 1)$. The error shrinks by a factor of $|\lambda_2/\lambda_1| = 2/5$ per step.

Why it works: write $\mathbf{v}_0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots$ in eigenvectors. Then $A^k\mathbf{v}_0 = c_1\lambda_1^k\mathbf{v}_1 + c_2\lambda_2^k\mathbf{v}_2 + \cdots = \lambda_1^k\big(c_1\mathbf{v}_1 + c_2(\lambda_2/\lambda_1)^k\mathbf{v}_2 + \cdots\big)$. If $|\lambda_1|$ is strictly the largest, every ratio $(\lambda_i/\lambda_1)^k$ shrinks to zero, leaving only the $\mathbf{v}_1$ direction. Renormalising each step stops the numbers overflowing.

Example 37. Procedure — power iteration (and PageRank)

Input: a square matrix A (for PageRank, the Google matrix G, or just a way to compute $G\mathbf{v}$); a tolerance ε; a maximum number of iterations.

Steps:

1. Start with any vector $\mathbf{v}$ having a component along the dominant eigenvector (for PageRank, the uniform vector $(1/n, \dots, 1/n)$).
2. Compute $\mathbf{w} = A\mathbf{v}$.
3. Renormalise: $\mathbf{w} \leftarrow \mathbf{w} / \|\mathbf{w}\|$ (for PageRank, divide by the sum of entries, which for G stays 1 automatically).
4. If $\|\mathbf{w} - \mathbf{v}\| < \varepsilon$, stop. Otherwise set $\mathbf{v} \leftarrow \mathbf{w}$ and go to step 2.
5. Estimate the eigenvalue with the **Rayleigh quotient** $\lambda \approx \mathbf{v}^{\mathsf{T}} A \mathbf{v}$ (for a unit vector $\mathbf{v}$); for PageRank it's 1 by construction.

Output: the dominant eigenvector $\mathbf{v}$ (the PageRank vector) and its eigenvalue. Convergence is geometric with ratio $|\lambda_2 / \lambda_1|$; for the Google matrix, $|\lambda_2| \le d$, so the error shrinks at least by 0.85 per iteration.

28.7.5. Worked example: ranking the tiny web

Before running code, predict. D has no incoming links, so it only gets teleport traffic: $r_D = (1-d)/n = 0.15/4 = 0.0375$. C receives links from three pages (including all of B's vote and all of D's) and should rank highest. A receives **all** of C's vote, so A should be close behind. B receives only half of A's vote. *Try it:* Rank the tiny web ; Rank the tiny web .

```python
import numpy as np

pages = ["A", "B", "C", "D"]
links = {"A": ["B", "C"], "B": ["C"], "C": ["A"], "D": ["C"]}
n = len(pages)
idx = {p: i for i, p in enumerate(pages)}

# Link matrix: column j spreads page j's vote equally over its out-links.
M = np.zeros((n, n))
for src, dests in links.items():
    for dst in dests:
        M[idx[dst], idx[src]] = 1 / len(dests)

d = 0.85                                  # damping factor
G = d * M + (1 - d) / n * np.ones((n, n))  # the "Google matrix"

r = np.full(n, 1 / n)                     # start: equal rank everywhere
for it in range(1, 101):
    r_new = G @ r
    change = np.abs(r_new - r).sum()
    r = r_new
    if change < 1e-10:
        break
print("iterations:", it)
for p in pages:
    print(p, round(r[idx[p]], 4))
```

```
iterations: 47
A 0.3725
B 0.1958
C 0.3941
D 0.0375
```

Check the steady-state equations by hand. Each page's rank is its teleport share plus d times the votes it receives:

$$r_A = 0.0375 + 0.85\, r_C = 0.0375 + 0.85(0.3941) \approx 0.3725 \checkmark$$
$$r_B = 0.0375 + 0.85 \cdot \tfrac{1}{2} r_A = 0.0375 + 0.425(0.3725) \approx 0.1958 \checkmark$$
$$r_D = 0.0375 \checkmark$$

and the four ranks sum to 1. The prediction holds: C first, A a close second (one very good inbound link can be worth as much as several mediocre ones), then B, then the unlinked D.

Computer-science connection: PageRank at web scale

The real link matrix is **sparse**: a page links to dozens of others, not billions, so M has a few dozen nonzeros per column. Power iteration never forms G (which would be dense); it computes $G\mathbf{r} = dM\mathbf{r} + \frac{1-d}{n}\mathbf{1}$ using the sparse M and the fact that $J\mathbf{r} = \mathbf{1}$ when $\mathbf{r}$ sums to 1. Each iteration costs $O(\text{number of links})$ and parallelises naturally: it's the canonical example in MapReduce and in graph-processing systems like Pregel. About 50–100 iterations suffice in practice, because the error shrinks by at least 0.85 per step ($0.85^{100} \approx 10^{-7}$).

Variations of the same eigenvector idea rank people in social networks (**eigenvector centrality**, Volume 3, Chapter 25), scientific papers by citations, players in tournaments, and the importance of code modules in a dependency graph. **Personalised PageRank** replaces the uniform teleport with a jump back to a user's own pages, which makes it a recommendation engine.



The $25,000,000,000 eigenvector

That was the title of a 2006 paper by Kurt Bryan and Tanya Leise explaining PageRank to undergraduates, the figure being roughly Google's market value at the time. The idea wasn't entirely new: in 1976 Gabriel Pinski and Francis Narin ranked scientific journals by an eigenvector of the citation matrix, and in 1953 the sociologist Leo Katz proposed a similar measure of social status. What Page and Brin added was the random-surfer interpretation, the damping trick that makes it well-posed on a messy real web, and the engineering to compute it over the entire web. "PageRank" is a pun: it ranks pages, and it was invented by Larry Page.



Grow your own web

The learner draws pages as nodes and links as arrows. The app animates a crowd of random surfers hopping along links (with occasional teleports), sizes each node by the fraction of surfers on it, and shows the power-iteration vector converging alongside. Sliders: damping factor d from 0 to 1 (watch rankings flatten towards uniform as $d \to 0$, and convergence slow as $d \to 1$). Challenges: add a "link farm" of pages that all point at one page and see how much rank it gains; create a dead-end page and a closed loop and observe what goes wrong with $d = 1$.

Try it: Grow your own web · PageRank with a damping slider

28.8. Exercises: powers, chains, and ranking

1. **(Recognition)** A Markov transition matrix has eigenvalues 1, 0.95 and 0.2. Another has 1, 0.5 and 0.1. Which converges to its steady state faster, and why?
2. **(Application)** Use diagonalisation to find a formula for $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^n$ (eigenvalues and eigenvectors from the earlier exercise).

3. **(Application)** A gym member who works out today works out tomorrow with probability 0.7; one who skips today works out tomorrow with probability 0.4. Find the long-run fraction of days spent working out. *Try it:* Check the gym chain .

4. **(Problem solving)** In the tiny web, add a link from B to D. Without computing, predict how the ranks of B, C and D change. Then explain why a page with no out-links (a "dangling node") breaks the column-sum property of M, and suggest a fix. *Try it:* Add the link B to D ; Add the link B → D .

▼ **Answers**

1. The second: its second-largest eigenvalue size is 0.5, so errors shrink by half each step, versus only 5% per step ($\times 0.95$) for the first. Spectral gap 0.5 versus 0.05.
2. $P = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, $D = \mathrm{diag}(3, 1)$, $P^{-1} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. $PD^nP^{-1} = \frac{1}{2}\begin{bmatrix} 3^n+1 & 3^n-1 \\ 3^n-1 & 3^n+1 \end{bmatrix}$. Check $n = 1$: $\frac{1}{2}(4, 2; 2, 4) = (2, 1; 1, 2)$ ✓.
3. States (work out, skip): $P = \begin{bmatrix} 0.7 & 0.4 \\ 0.3 & 0.6 \end{bmatrix}$. Steady state: $-0.3w + 0.4s = 0$, so $w = \frac{4}{3}s$; with $w + s = 1$, $s = 3/7$, $w = 4/7 \approx 0.571$.
4. B's vote is now split between C and D, so C loses rank and D gains (it finally has an inbound link). A depends entirely on C, so A falls too, and B, which depends on A, falls slightly. Running the code with the extra link confirms it: A 0.3426, B 0.1831, C 0.3590, D 0.1153. A dangling page's column in M is all zeros, so the column sums to 0, not 1, and probability "leaks" out each step. Standard fix: treat a dangling page as linking to every page (replace its column with $1/n$ everywhere), i.e. the surfer teleports from dead ends.

28.9. Symmetric matrices and orthogonal eigenvectors

28.9.1. The observation

In the exercise above, $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ had eigenvectors $(1, 1)$ and $(1, -1)$, which are **perpendicular**. That's no accident. The matrix is **symmetric** ($A^{\mathsf{T}} = A$, Chapter 27), and symmetric matrices are the best-behaved matrices in linear algebra.



The spectral theorem

If A is a real symmetric $n \times n$ matrix, then:

1. all its eigenvalues are real (no rotation hiding inside);
2. eigenvectors for different eigenvalues are orthogonal;
3. it always has n orthonormal eigenvectors, so it can be diagonalised as

$$A = Q \Lambda Q^{\mathsf{T}},$$

where Q is an orthogonal matrix (orthonormal eigenvector columns, so $Q^{-1} = Q^{\mathsf{T}}$) and Λ (capital lambda) is diagonal.

Geometrically: **every symmetric matrix is a pure stretch along some set of perpendicular axes**. Rotate to those axes (Q^{T}), stretch (Λ), rotate back (Q). No shear, no rotation.

28.9.2. Why the eigenvectors are perpendicular

The proof is short and uses only the transpose rule $\mathbf{u} \cdot (A\mathbf{v}) = \mathbf{u}^{\mathsf{T}} A \mathbf{v} = (A^{\mathsf{T}}\mathbf{u})^{\mathsf{T}}\mathbf{v}$. Suppose $A\mathbf{u} = \lambda \mathbf{u}$ and $A\mathbf{v} = \mu \mathbf{v}$ with $\lambda \neq \mu$. Compute $\mathbf{u}^{\mathsf{T}} A \mathbf{v}$ two ways:

$$\begin{aligned} &\mathbf{u}^{\mathsf{T}}(A\mathbf{v}) = \mathbf{u}^{\mathsf{T}}(\mu \mathbf{v}) = \mu\,(\mathbf{u} \cdot \mathbf{v}), \\ &(\mathbf{u}^{\mathsf{T}} A)\mathbf{v} = (A^{\mathsf{T}}\mathbf{u})^{\mathsf{T}}\mathbf{v} = (A\mathbf{u})^{\mathsf{T}}\mathbf{v} = \lambda\,(\mathbf{u} \cdot \mathbf{v}). \qquad (\text{symmetry: } A^{\mathsf{T}} = A) \end{aligned}$$

The two are equal, so $(\lambda - \mu)(\mathbf{u} \cdot \mathbf{v}) = 0$. Since $\lambda \neq \mu$, we must have $\mathbf{u} \cdot \mathbf{v} = 0$. ∎

Symmetric matrices turn up constantly: covariance matrices (next section), the $A^{\mathsf{T}}A$ of least squares, graph Laplacians, the Hessian of second derivatives (Chapter 20), and the stiffness matrices of engineering. Whenever you have one, use `np.linalg.eigh`, which exploits the symmetry.

28.10. Principal component analysis

28.10.1. The problem

You've measured 500 people's height and arm span. The two numbers are strongly correlated: tall people have long arms. Plotted, the data form an elongated, tilted cloud. Is there a single number that captures most of what distinguishes one person from another? More generally, you have data with 768 features per item (embeddings, again). Can you find a handful of directions that capture most of the variation, to visualise, compress, or denoise it?

28.10.2. Intuition: find the long axis of the cloud

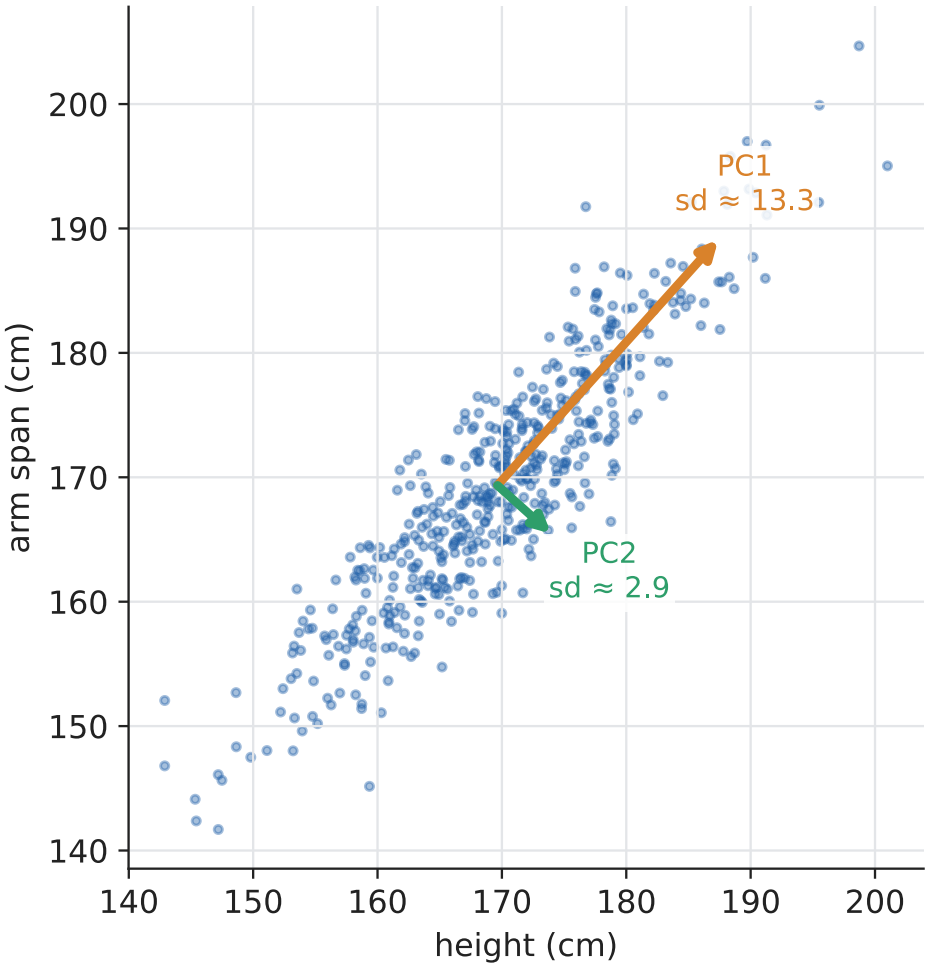


Figure 81. Height and arm span for 500 simulated people. The first principal component (PC1, orange) points along the cloud's long axis; the second (PC2, green) is perpendicular to it. Arrows are drawn two standard deviations long.

The cloud is long in one direction and thin in the other. If you projected every point onto the long axis (Chapter 26), you'd keep most of the spread between people and lose only the small sideways scatter. That long axis is the **direction of maximum variance**, and finding it is principal component analysis.

28.10.3. Why it works: variance along a direction is a quadratic form

Centre the data (subtract the mean of each feature), so the cloud sits at the origin. Put the N centred data points as rows of a matrix X ($N \times p$ for p features). The variance of the data projected onto a unit direction $\mathbf{u}$ is the average of the squared projections $(\mathbf{x}_i \cdot \mathbf{u})^2$:

$$\mathbf{Var}(\mathbf{u}) = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i^{\mathsf{T}}\mathbf{u})^2 = \frac{1}{N-1}\|X\mathbf{u}\|^2 = \mathbf{u}^{\mathsf{T}}\underbrace{\left(\tfrac{1}{N-1}X^{\mathsf{T}}X\right)}_{C}\mathbf{u}.$$

The matrix C is the **covariance matrix** ($p \times p$, symmetric). Its diagonal entries are the variances of each feature; its off-diagonal entries are the covariances between pairs of features (Volume 3, Chapter 31). Now write $C = Q\Lambda Q^{\mathsf{T}}$ (spectral theorem) and let $\mathbf{y} = Q^{\mathsf{T}}\mathbf{u}$, the direction expressed in eigen-coordinates (still a unit vector, since Q^{T} is a rotation):

$$\mathrm{Var}(\mathbf{u}) = \mathbf{u}^\mathsf{T} Q \Lambda Q^\mathsf{T} \mathbf{u} = \mathbf{y}^\mathsf{T} \Lambda \mathbf{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \cdots + \lambda_p y_p^2.$$

With $y_1^2 + \cdots + y_p^2 = 1$, this is a weighted average of the eigenvalues. It's largest when all the weight goes on the largest eigenvalue: $\mathbf{y} = (1, 0, \ldots, 0)$, i.e. $\mathbf{u} =$ the top eigenvector. So:

- **The direction of maximum variance is the eigenvector of the covariance matrix with the largest eigenvalue**, and that eigenvalue **is** the variance along it.
- The next-best direction perpendicular to the first is the second eigenvector, and so on.
- The eigenvalues add up to the total variance (the trace of C), so $\lambda_k / \sum \lambda_i$ is the **fraction of variance explained** by component k.



Covariance matrix, principal components

- The **covariance matrix** of centred data X (rows = observations) is $C = \frac{1}{N-1} X^\mathsf{T} X$. Entry c_{jk} is the covariance of features j and k; the diagonal holds variances. It's symmetric with non-negative eigenvalues.
- The **principal components** (PCs) are the eigenvectors of C, sorted by decreasing eigenvalue: orthogonal directions of decreasing variance.
- The **scores** of a data point are its coordinates along the PCs: XQ_k for the top k components. Keeping $k < p$ of them is **dimensionality reduction**.

Example 38. Procedure — principal component analysis

Input: a data matrix X with N rows (observations) and p columns (features); the number k of components to keep.

Steps:

1. **Centre:** subtract each column's mean. (If features have different units, e.g. centimetres and kilograms, also divide each column by its standard deviation, i.e. **standardise**.)
2. **Covariance:** $C = \frac{1}{N-1} X^\mathsf{T} X$.
3. **Eigen-decompose** the symmetric C (`np.linalg.eigh`), and sort eigenvalues from largest to smallest, reordering the eigenvector columns to match.
4. **Choose k:** look at the fraction of variance explained; keep enough components to reach, say, 90–95%, or look for an "elbow" where eigenvalues drop off.
5. **Project:** scores $= XQ_k$, where Q_k holds the top k eigenvectors.

Output: the principal directions, their variances, and each observation's k-dimensional scores.

28.10.4. Worked example: height and arm span

```python
import numpy as np

rng = np.random.default_rng(1)
# 500 people: height (cm) and arm span (cm), strongly correlated.
height = rng.normal(170, 10, 500)
span = height + rng.normal(0, 4, 500)
X = np.column_stack([height, span])

Xc = X - X.mean(axis=0)                 # 1. centre each column
C = Xc.T @ Xc / (len(X) - 1)            # 2. covariance matrix (2 x 2)
vals, vecs = np.linalg.eigh(C)          # 3. eigen-decomposition (symmetric)
order = np.argsort(vals)[::-1]          #    sort largest first
vals, vecs = vals[order], vecs[:, order]
vecs = vecs * np.sign(vecs[0])          #    (fix the arbitrary sign of each axis)

print("covariance:\n", C.round(1))
print("variances along principal axes:", vals.round(1))
print("fraction explained:", (vals / vals.sum()).round(3))
print("first principal direction:", vecs[:, 0].round(3))

scores = Xc @ vecs[:, :1]               # 4. project onto the top axis
print(scores.shape)
```

Try it: PCA on height and arm span .

```
covariance:
 [[ 83.5  83.5]
 [ 83.5 101.3]]
variances along principal axes: [176.4   8.4]
fraction explained: [0.954 0.046]
first principal direction: [0.669 0.744]
(500, 1)
```

Interpret each line. The covariance matrix says height varies with variance 83.5 (standard deviation about 9.1 cm), arm span slightly more (101.3, because it adds measurement noise), and they covary strongly (83.5: arm span is height plus independent noise, so their covariance is approximately height's own variance). The first principal component, $(0.669, 0.744)$, points roughly along the diagonal: "overall size", with both measurements increasing together. It carries 95.4% of the variance. The second, perpendicular to it, is "arms long or short for your height", with only 4.6%. So one number per person, their score on PC1, captures almost everything: two dimensions have been reduced to one with little loss. The trace checks out too: $83.5 + 101.3 = 184.8 \approx 176.4 + 8.4$.

28.10.5. Choosing a method, and when PCA misleads

PCA finds **linear** structure. It's the right first tool for correlated numeric features, for visualising high-dimensional data in 2D, and for compressing before another algorithm. Know its limits:



PCA failure modes

- **Forgetting to centre** makes the first "component" point at the mean rather than along the spread.
- **Mixed units.** Measure height in millimetres instead of centimetres and its variance grows 100-fold, so PC1 becomes "height" purely because of units. Standardise features with different units.
- **Variance is not importance.** The direction that separates two classes (say, healthy versus ill) may have **small** variance and be discarded. PCA is **unsupervised**: it never looks at labels.
- **Curved structure.** Data on a curve or a "Swiss roll" isn't captured by straight axes. Nonlinear methods (t-SNE, UMAP, autoencoders) exist for that, at the cost of interpretability.
- **Outliers** inflate variance in their own direction and can hijack a component.

Try it: See mixed units hijack PC1 .

28.10.6. Where you'd use this

- **Everyday: why clothes come in S, M and L.** Our height-and-arm-span example is the reason a single "size" works at all: body measurements are so strongly correlated that one number, essentially the score on PC1 ("overall size"), captures most of the variation. Garment designers run PCA on thousands of body scans to choose size charts that fit the most people with the fewest sizes.
- **Science:** in a 2008 study of about 1,400 Europeans genotyped at about 200,000 genetic markers, a plot of each person's scores on the first two principal components reproduced a recognisable map of Europe; geography was the direction of greatest genetic variation. In finance, three principal components of interest-rate movements, interpreted as "level", "slope" and "curvature" of the yield curve, typically explain well over 90% of the day-to-day variation, so banks hedge risk against those three numbers instead of dozens of rates. Climate scientists call principal components of weather maps **empirical orthogonal functions**; the leading one in the tropical Pacific is the El Niño pattern.
- **Computing:** "eigenfaces" (1991) recognised faces by projecting photos onto the top principal components of a face dataset; PCA is also the standard first step before clustering or plotting any high-dimensional data, as the next sidebar describes.

Computer-science connection: dimensionality reduction for embeddings

Embedding vectors (Chapter 26) typically have hundreds or thousands of dimensions, but their **effective** dimension is often much lower: PCA on a collection of sentence embeddings usually shows the variance concentrated in a fraction of the components. That's useful in several ways:

- **Visualisation:** project to the top 2 or 3 PCs to plot clusters of documents (with the caveats from Chapter 26 about high-dimensional distances).
- **Compression:** storing 128 PCA scores instead of 768 raw components cuts a vector index's memory six-fold, usually with a modest loss in retrieval quality. Vector databases combine this with quantisation.
- **Whitening and de-biasing:** subtracting the mean and the top one or two components often **improves** cosine-similarity search, because those components encode features shared by all texts (the narrow "cone" that makes unrelated texts score 0.6). Removing them spreads the vectors out.
- **Faster search:** searching in a lower-dimensional space is cheaper per comparison.

Some modern embedding models are trained so that their **first** k components already form a good k-dimensional embedding ("Matryoshka" embeddings), building the PCA-like ordering into the model itself.



Rotate the cloud

A scatter plot of 2D data with a draggable line through the centre. As the learner rotates the line, the app projects every point onto it and shows the variance of the projections as a live number and a dial. The learner hunts for the maximum, then clicks "reveal" to see PC1 drawn from the eigenvector of the covariance matrix, landing exactly on their best line. Further sliders change the correlation and the noise, so the learner sees the eigenvalue ratio (variance explained) track how elongated the cloud is.

Try it: Rotate the cloud

28.11. The singular value decomposition

28.11.1. The problem

Eigenvectors have a frustrating limitation: they only exist for **square** matrices (a 3×2 matrix takes 2D vectors to 3D vectors, so "$A\mathbf{v}$ is a multiple of $\mathbf{v}$" makes no sense), and even square matrices may lack enough of them. Is there a decomposition that works for **every** matrix, of any shape?

28.11.2. Intuition: rotate, scale, rotate

There is, and it's the most useful factorisation in applied mathematics. Watch what a 2×2 matrix does to the unit circle: it always produces an **ellipse**. An ellipse has perpendicular axes. So some pair of perpendicular input directions (on the circle) must map to the perpendicular axes of the ellipse. That gives a three-step description of **any** matrix:

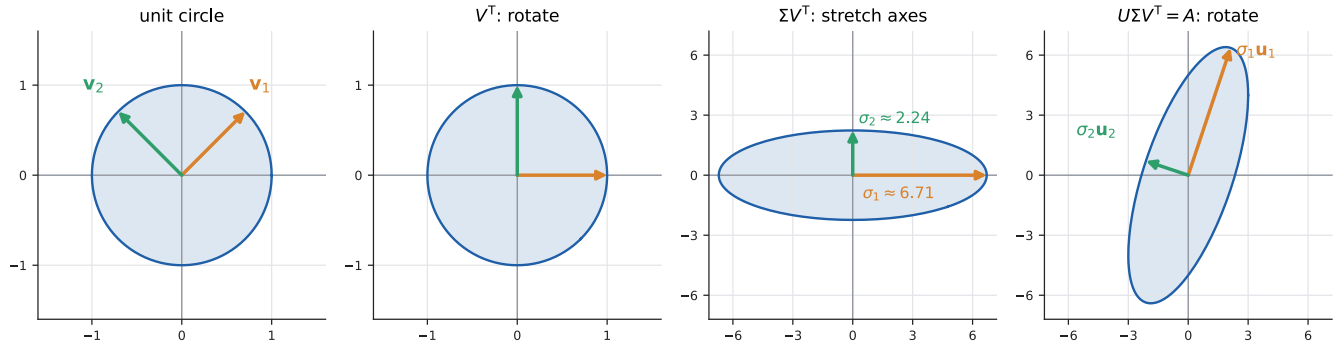


Figure 82. Any matrix maps the unit circle to an ellipse, in three steps: rotate (or reflect) the input with V^T, stretch along the coordinate axes by the singular values with Σ, then rotate into the output position with U. Shown for the worked example below, $A = \begin{bmatrix} 3 & 0 \\ 4 & 5 \end{bmatrix}$, with $\mathbf{v}_1 = (1,1)/\sqrt{2}$ (orange) and $\mathbf{v}_2 = (-1,1)/\sqrt{2}$ (green; its sign is flipped from the worked example so both outer steps are rotations), stretched by $\sigma_1 \approx 6.71$ and $\sigma_2 \approx 2.24$. The first two panels are drawn at a larger scale than the last two.

$$\boxed{A = U \Sigma V^\mathsf{T}}$$

- V ($n \times n$, orthogonal): its columns $\mathbf{v}_i$ are the **right singular vectors**, the perpendicular input directions.
- Σ (capital sigma, $m \times n$, diagonal): the **singular values** $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$, the stretch factors.
- U ($m \times m$, orthogonal): its columns $\mathbf{u}_i$ are the **left singular vectors**, the output directions (the ellipse's axes).

And $A\mathbf{v}_i = \sigma_i \mathbf{u}_i$: each input axis goes to an output axis, scaled. This is the eigenvector idea generalised: you're allowed **different** directions on the input and output sides, and in exchange it works for every matrix.



Singular value decomposition

Every real $m \times n$ matrix can be written $A = U\Sigma V^\mathsf{T}$ with U, V orthogonal and Σ diagonal with non-negative entries $\sigma_1 \geq \sigma_2 \geq \cdots$ (the **singular values**). The singular values are the square roots of the eigenvalues of the symmetric matrix $A^\mathsf{T}A$; the columns of V are its eigenvectors. The number of nonzero singular values is the **rank** of A. For a symmetric matrix with non-negative eigenvalues, the SVD and the eigendecomposition coincide.

28.11.3. Worked example: a 2×2 SVD by hand

Find the singular values of $A = \begin{bmatrix} 3 & 0 \\ 4 & 5 \end{bmatrix}$. *Try it:* SVD as rotate–stretch–rotate ; Circle to ellipse, step by step .

$$A^{\mathsf{T}}A = \begin{bmatrix} 3 & 4 \\ 0 & 5 \end{bmatrix}\begin{bmatrix} 3 & 0 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix},$$

$$\text{char. eq.: } \lambda^2 - 50\lambda + (625 - 400) = \lambda^2 - 50\lambda + 225 = (\lambda - 45)(\lambda - 5) = 0,$$

$$\sigma_1 = \sqrt{45} = 3\sqrt{5} \approx 6.708, \qquad \sigma_2 = \sqrt{5} \approx 2.236.$$

Eigenvectors of $A^{\mathsf{T}}A$ (same pattern as the $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ exercise): $\mathbf{v}_1 = (1,1)/\sqrt{2}$, $\mathbf{v}_2 = (1,-1)/\sqrt{2}$. The output directions are $\mathbf{u}_i = A\mathbf{v}_i/\sigma_i$:

$$\mathbf{u}_1 = \frac{1}{\sqrt{45}} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 3 \\ 9 \end{bmatrix} = \frac{1}{\sqrt{10}}\begin{bmatrix} 1 \\ 3 \end{bmatrix}, \qquad \mathbf{u}_2 = \frac{1}{\sqrt{5}} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 3 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{10}}\begin{bmatrix} 3 \\ -1 \end{bmatrix}.$$

Checks: $\mathbf{u}_1 \cdot \mathbf{u}_2 = (3-3)/10 = 0$, both have length 1, and $\sigma_1\sigma_2 = \sqrt{225} = 15 = |\det A|$ (area scaling is the product of the stretches). The unit circle becomes an ellipse with semi-axes 6.71 along $(1,3)$ and 2.24 along $(3,-1)$.

28.11.4. Low-rank approximation

Multiply out $U\Sigma V^{\mathsf{T}}$ column by column and you get a sum of simple pieces:

$$A = \sigma_1\mathbf{u}_1\mathbf{v}_1^{\mathsf{T}} + \sigma_2\mathbf{u}_2\mathbf{v}_2^{\mathsf{T}} + \cdots + \sigma_r\mathbf{u}_r\mathbf{v}_r^{\mathsf{T}}.$$

Each term $\mathbf{u}_i\mathbf{v}_i^{\mathsf{T}}$ (column times row, an **outer product**) is a rank-1 matrix, and the terms are ordered by importance σ_i. Keeping only the first k terms gives a rank-k matrix A_k. The **Eckart–Young theorem** says this truncation is the **best possible** rank-k approximation to A: no other rank-k matrix is closer (in the sense of summed squared entry differences). And storing it takes only $k(m+n+1)$ numbers instead of mn.

Example 39. Procedure — low-rank approximation by SVD

Input: an $m \times n$ matrix A (an image, a ratings table, a term–document matrix) and a target rank k.

Steps:

1. Compute the SVD: `U, s, Vt = np.linalg.svd(A, full_matrices=False)`.
2. Inspect the singular values `s` (plot them on a log scale) to choose k: look for where they drop off.
3. Keep the first k columns of U, the first k singular values and the first k rows of V^{T}.
4. Reconstruct $A_k = U_k\,\operatorname{diag}(s_k)\,V_k^{\mathsf{T}}$ if you need the matrix itself, or use the factors directly as compact representations.

Output: A_k, the best rank-k approximation, with relative error $\sqrt{\sum_{i>k}\sigma_i^2}\,/\,\sqrt{\sum_i \sigma_i^2}$.

28.11.5. Worked example: compressing an image

A greyscale image is just a matrix of brightness values. Here's a synthetic 200 × 200 image (smooth waves plus a bright disc plus noise) and its low-rank approximations: *Try it:* Compress the image yourself.

PYTHON

```
import numpy as np

# A synthetic 200 x 200 greyscale "image": smooth shapes plus a little noise.
y, x = np.mgrid[0:200, 0:200] / 200.0
img = (np.sin(6 * x) * np.cos(4 * y)
       + ((x - 0.6) ** 2 + (y - 0.4) ** 2 < 0.05)
       + 0.05 * np.random.default_rng(0).standard_normal((200, 200)))

U, s, Vt = np.linalg.svd(img)
print("largest singular values:", s[:6].round(1))

for k in [1, 5, 20, 50]:
    approx = U[:, :k] @ np.diag(s[:k]) @ Vt[:k, :]       # rank-k approximation
    err = np.linalg.norm(img - approx) / np.linalg.norm(img)
    stored = k * (200 + 200 + 1)
    print(f"rank {k:2d}: relative error {err:.3f}, numbers stored {stored:6d} of 40000")
```

```
largest singular values: [113.5  68.4  18.1  10.5   7.3   5.7]
rank  1: relative error 0.544, numbers stored    401 of 40000
rank  5: relative error 0.114, numbers stored   2005 of 40000
rank 20: relative error 0.071, numbers stored   8020 of 40000
rank 50: relative error 0.050, numbers stored  20050 of 40000
```

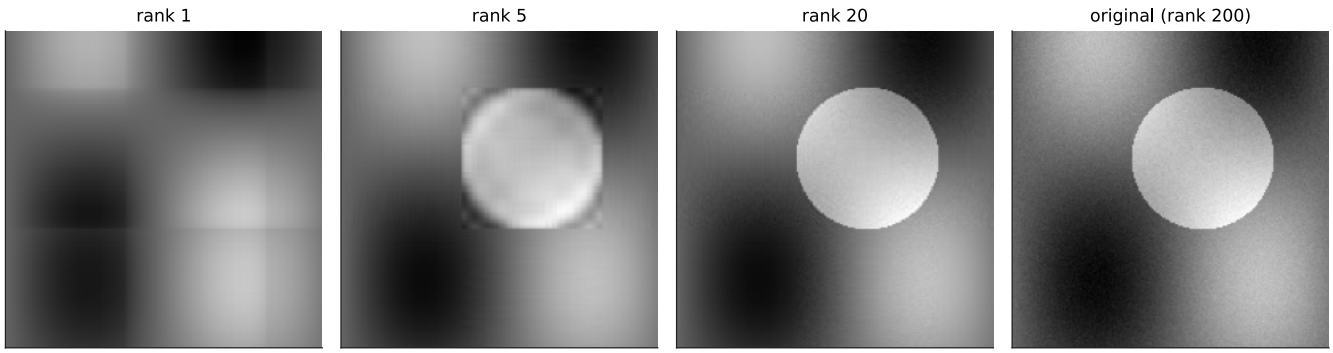


Figure 83. Rank-k approximations of a 200 × 200 image. Rank 5 already captures the waves and the disc; rank 20 is hard to tell from the original; the rest is mostly noise.

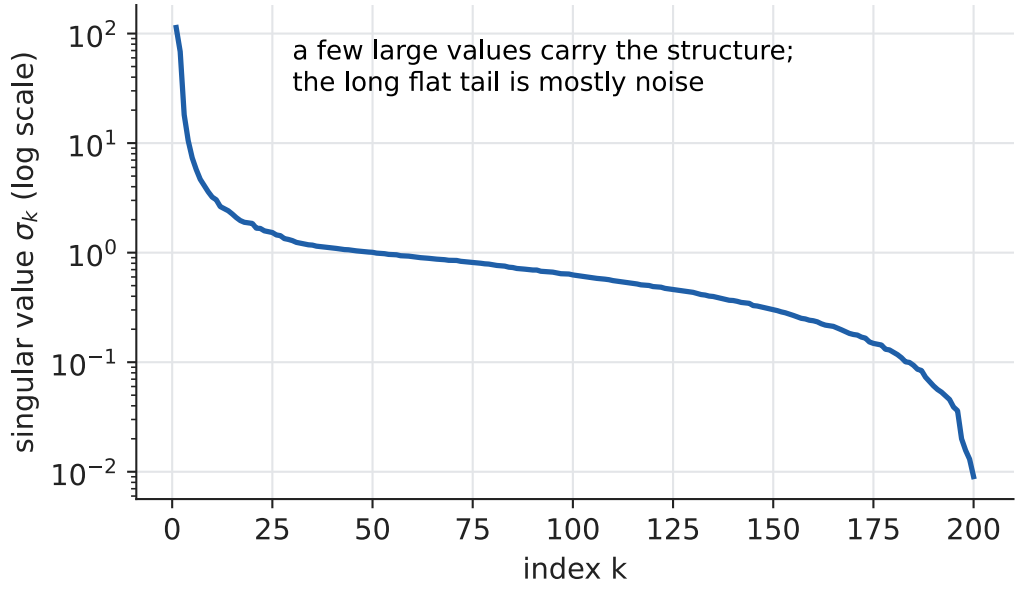


Figure 84. Singular values of the image on a log scale. A few large values carry the structure; the long, flat tail is mostly noise.

The **wave** part of the image, $\sin(6x)\cos(4y)$, is a product of a function of x and a function of y: exactly rank 1. That's why the first singular value is so dominant. The disc needs a few more terms (its edges are curved, not aligned with rows and columns). Beyond about rank 20 the remaining singular values are noise, and dropping them **improves** the image by denoising it. Real photographs aren't this cooperative, which is why JPEG uses a different transform (the discrete cosine transform), but the idea of "keep the big coefficients, drop the small ones" is the same.

28.11.6. Latent semantic indexing: SVD in information retrieval

Recall the blind spot of bag-of-words search (Chapter 26): a query for "car" scores zero against a document that only says "automobile". **Latent semantic indexing** (LSI, also called latent semantic analysis), introduced by Deerwester, Dumais and colleagues in 1990, fixes this with the SVD. *Try it:* LSI concept space .

Build the **term–document matrix** A: one row per term, one column per document. Its rank-k approximation forces the documents into a k-dimensional "concept" space. Terms that co-occur across documents ("car", "automobile" and "engine" all appear alongside each other) end up sharing a concept direction, so documents using **different words for the same thing** get similar concept vectors.

PYTHON

```python
import numpy as np

# Term-document matrix: rows are terms, columns are documents.
terms = ["car", "automobile", "engine", "flower", "petal"]
#              d1  d2  d3  d4  d5
A = np.array([[1,  0,  1,  0,  0],    # car
              [0,  1,  1,  0,  0],    # automobile
              [1,  1,  1,  0,  0],    # engine
              [0,  0,  0,  1,  1],    # flower
              [0,  0,  0,  1,  0]],   # petal
             dtype=float)
q = np.array([1, 0, 0, 0, 0], dtype=float)     # query: "car"

U, s, Vt = np.linalg.svd(A)
k = 2                                           # keep two "concepts"
Uk, sk = U[:, :k], s[:k]
docs_k = Vt[:k, :].T * sk                       # documents in concept space
q_k = q @ Uk                                    # query in concept space

def cos(a, b):
    return a @ b / (np.linalg.norm(a) * np.linalg.norm(b))

for j in range(5):
    print(f"d{j+1}: raw {cos(q, A[:, j]):.2f}   latent {cos(q_k, docs_k[j]):.2f}")
```

```
d1: raw 0.71   latent 1.00
d2: raw 0.00   latent 1.00
d3: raw 0.58   latent 1.00
d4: raw 0.00   latent 0.00
d5: raw 0.00   latent 0.00
```

Document d2 never mentions "car" and scores 0.00 on raw cosine similarity. In the 2-concept space it scores as high as the documents that do, because "automobile" and "engine" share the vehicle concept with "car". The flower documents stay at zero. (With only two clear topics, rank 2 sorts everything perfectly; real collections use k of a few hundred.)

28.11.7. Where you'd use this

- **Everyday:** the worked example above is the arithmetic of image compression: the rank-20 picture keeps 8020 of 40,000 numbers, 20% of the storage, with about 7% error. Removing hiss from an old recording or noise from a low-light photo uses the same move: keep the large singular values (the signal) and drop the flat tail (mostly noise).
- **Science and engineering:** to line up two 3D scans of the same object, or compare two shapes of the same protein, you need the best rotation taking one set of points onto the other. The standard answer (the **Kabsch algorithm**) takes the SVD of a 3×3 matrix built from the points and reads the rotation off U and V. Medical imaging, seismology and climate science all use the SVD to separate a few strong patterns from noisy measurements, and the ratio σ_1/σ_n (the condition number) tells an engineer whether a measurement system can be trusted to invert (Chapter 32).
- **Computing:** recommendation, search, model compression and least squares, collected in the next sidebar.

Computer-science connection: the SVD in the wild

- **Recommender systems.** A users × items ratings matrix is mostly empty. Approximating it with a low-rank product UV^T (each user and each item gets a short "taste" vector) predicts the missing ratings; this **matrix factorisation** approach was central to the winning entries of the 2006–2009 Netflix Prize.
- **Embeddings.** LSI was an early ancestor of today's embeddings. Word2vec-style word vectors were later shown to be closely related to factorising a word co-occurrence matrix.
- **Model compression.** Large neural-network weight matrices are often approximately low-rank. **LoRA** (low-rank adaptation) fine-tunes a language model by learning a small rank-k update BA to each weight matrix instead of changing all its entries, cutting the trainable parameters by orders of magnitude.
- **PCA via SVD.** In practice PCA is computed from the SVD of the centred data matrix, never by forming $X^\mathsf{T}X$: the right singular vectors of X are the principal directions and $\sigma_i^2/(N-1)$ are the variances, and skipping $X^\mathsf{T}X$ avoids squaring the condition number (Chapter 32).
- **Least squares and pseudo-inverses.** The SVD gives the most stable way to solve over-determined and rank-deficient systems (`np.linalg.lstsq` uses it).



Choosing between eigendecomposition, PCA and SVD

- **Square matrix applied repeatedly** (dynamics, Markov chains, recurrences, PageRank): eigenvalues and eigenvectors. The largest eigenvalue governs the long run.
- **Symmetric matrix** (covariance, Laplacian, Hessian): eigendecomposition with `eigh`; orthogonal eigenvectors guaranteed.
- **Data cloud, directions of most variation:** PCA, computed via the SVD of the centred data.
- **Any rectangular matrix, compression, denoising, rank, least squares:** the SVD.

Compress it yourself

The learner uploads a photo (or picks one). The app converts it to greyscale, computes its SVD and shows a slider for the rank k, redrawing the rank-k image live alongside a plot of the singular values with the cutoff marked, the storage cost, and the relative error. A second panel shows the individual rank-1 "layers" $\sigma_i \mathbf{u}_i \mathbf{v}_i^{\mathsf{T}}$ for $i = 1, 2, 3, \ldots$, so the learner sees the image being built from stripes. A third shows the unit circle turning into an ellipse for a user-chosen 2×2 matrix, step by step through V^{T}, Σ and U.

Try it: Unit square through V^{T}, Σ, U · Compress it yourself

28.12. Exercises: symmetric matrices, PCA, and SVD

1. **(Recognition)** A covariance matrix for three features has eigenvalues 9, 0.9 and 0.1. What fraction of the variance does the first principal component explain? How many components would you keep?

2. **(Application)** Find the principal directions of data with covariance matrix $C = \begin{bmatrix} 5 & 2 \\ 2 & 2 \end{bmatrix}$ and the variance along each.

3. **(Application)** Find the singular values of $\begin{bmatrix} 2 & 0 \\ 0 & -3 \end{bmatrix}$. Why aren't they just the eigenvalues?

4. **(Problem solving)** A 1000 × 800 matrix has singular values that fall from 500 to about 1 by index 30 and stay near 1 after that. Estimate the storage saving of a rank-30 approximation and the relative error, and say what the flat tail probably represents.

▼ **Answers**

1. Total variance $9 + 0.9 + 0.1 = 10$, so PC1 explains $9/10 = 90\%$. PC1 and PC2 together explain 99%. Keep one component if 90% suffices, two for 99%; the third (1%) is probably noise.

2. Trace 7, determinant $10 - 4 = 6$: $\lambda^2 - 7\lambda + 6 = (\lambda - 6)(\lambda - 1) = 0$. For 6: $C - 6I = \begin{bmatrix} -1 & 2 \\ 2 & -4 \end{bmatrix}$, $x = 2y$, direction $(2, 1)/\sqrt{5}$. For 1: $C - I = \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}$, $y = -2x$, direction $(1, -2)/\sqrt{5}$. Perpendicular ✓ ($2 - 2 = 0$). Variances 6 and 1; PC1 explains $6/7 \approx 86\%$.

3. $A^{\mathsf{T}}A = \mathrm{diag}(4, 9)$, so $\sigma = 3, 2$ (sorted). Eigenvalues are 2 and -3; singular values are always non-negative stretch factors, and the sign (a flip) is absorbed into U or V. Singular values are the absolute values of the eigenvalues here because the matrix is symmetric.

4. Full storage: $800\,000$ numbers. Rank 30: $30(1000 + 800 + 1) = 54\,030$, about 6.8% of the original. Relative error: $\sqrt{\sum_{i>30} \sigma_i^2} / \sqrt{\sum \sigma_i^2}$; the tail has about 770 values near 1, contributing roughly 770, while the top 30 contribute at least $500^2 = 250\,000$, so the error is roughly $\sqrt{770/250\,000} \approx 0.055$ or less, about 5%. The flat tail is most likely noise: structureless variation spread evenly across all directions.

28.13. Summary

- An **eigenvector** of A is a direction A only stretches: $A\mathbf{v} = \lambda\mathbf{v}$. The **eigenvalue** λ is the stretch factor. Find eigenvalues from $\det(A - \lambda I) = 0$ (for 2×2: $\lambda^2 - (\operatorname{tr} A)\lambda + \det A = 0$), then eigenvectors from $(A - \lambda I)\mathbf{v} = \mathbf{0}$.
- Eigenvectors can be complex (rotations) or too few (shears). In code, use `eig`, or `eigh` for symmetric matrices, never polynomial roots.
- **Change of basis**: P converts eigen-coordinates to standard ones. **Diagonalisation** $A = PDP^{-1}$ makes powers easy: $A^n = PD^nP^{-1}$. The largest eigenvalue governs the long run.
- **Fibonacci**: the matrix $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ has eigenvalues φ, ψ, giving Binet's formula; repeated squaring gives exact values in $O(\log n)$ steps.
- A **Markov chain**'s steady state is the eigenvector with eigenvalue 1; the second eigenvalue sets the convergence speed. **PageRank** is the steady state of a damped random surfer, computed by **power iteration**.
- **Symmetric matrices** have real eigenvalues and orthonormal eigenvectors: $A = Q\Lambda Q^{\mathsf{T}}$, a pure stretch along perpendicular axes.
- **PCA**: the eigenvectors of the covariance matrix are the directions of maximum variance; eigenvalues are the variances.
- The **SVD** $A = U\Sigma V^{\mathsf{T}}$ (rotate, scale, rotate) works for every matrix. Truncating it gives the best low-rank approximation, used for image compression, latent semantic indexing, recommenders and model compression.

28.14. Interactive tools

Browser tools on the companion website that go with this chapter. Each example link in the chapter opens one of them with that example already set up.

- Matrix calculator
- Matrix transformations
- Language sandbox
- Eigenvectors at work: Markov chains, PCA, SVD
- Graph algorithms

28.15. Mastery check

1. Show that $(1, 2)$ is an eigenvector of $\begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix}$ and state its eigenvalue. Then find the other eigenvalue using the trace, without solving a quadratic, and confirm it with the determinant.
2. Find all eigenvalues and eigenvectors of $\begin{bmatrix} 5 & -2 \\ 1 & 2 \end{bmatrix}$.
3. For the matrix in question 2, write $A = PDP^{-1}$ and use it to express $A^{10}(3, 2)$ in terms of 4^{10} and 3^{10}. What is $A^{10}(1, 1)$?
4. Explain in two or three sentences why a 2D rotation by $30°$ has no real eigenvectors, and what its eigenvalues are.
5. A Markov chain on two states has transition matrix $\begin{bmatrix} 0.8 & 0.3 \\ 0.2 & 0.7 \end{bmatrix}$ (column-stochastic). Find its steady state and its second eigenvalue, and say how quickly it converges.
6. In power iteration, why does the starting vector not matter (almost), and what makes convergence slow?
7. Data have covariance matrix $\begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$. What are the principal components and the fraction of variance explained by the first? Describe the shape of the data cloud.
8. State what each factor in $A = U\Sigma V^{\mathsf{T}}$ does geometrically, and explain how truncating it compresses a matrix.

Decision rule: All correct without hesitation → you've completed Part VII; move on to Volume 3, Chapter 29 (or Volume 3, Chapter 31 , where covariance and regression use this chapter directly). Stuck on 1–2 or 4 → reread Section 28.1 and Section 28.2. Stuck on 3 → reread Section 28.4. Stuck on 5–6 → reread Section 28.6 and Section 28.7. Stuck on 7–8 → reread Section 28.9, Section 28.10 and Section 28.11.

▼ **Mastery check answers**

1. $A(1, 2) = (2 + 2,\ 2 + 6) = (4, 8) = 4(1, 2)$, so $(1, 2)$ is an eigenvector with $\lambda = 4$. The trace is $2 + 3 = 5 = \lambda_1 + \lambda_2$, so the other eigenvalue is $5 - 4 = 1$. Determinant check: $2 \cdot 3 - 1 \cdot 2 = 4 = 4 \times 1$ ✓. (Its eigenvector: $A - I = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$ gives $(1, -1)$.)
2. Trace 7, determinant $10 + 2 = 12$: $\lambda^2 - 7\lambda + 12 = (\lambda - 4)(\lambda - 3) = 0$. For 4: $A - 4I = \begin{bmatrix} 1 & -2 \\ 1 & -2 \end{bmatrix}$, $x = 2y$, $\mathbf{v}_1 = (2, 1)$. Check: $A(2, 1) = (10 - 2, 2 + 2) = (8, 4)$ ✓. For 3: $A - 3I = \begin{bmatrix} 2 & -2 \\ 1 & -1 \end{bmatrix}$, $x = y$, $\mathbf{v}_2 = (1, 1)$. Check: $A(1, 1) = (3, 3)$ ✓.

3. $P = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$, $D = \mathbf{diag}(4, 3)$, $\det P = 1$, $P^{-1} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$. Eigen-coordinates of $(3, 2)$: $P^{-1}(3, 2) = (3 - 2,\ -3 + 4) = (1, 1)$, i.e. $(3, 2) = 1 \cdot (2, 1) + 1 \cdot (1, 1)$. So $A^{10}(3, 2) = 4^{10}(2, 1) + 3^{10}(1, 1) = (2\,097\,152 + 59\,049,\ 1\,048\,576 + 59\,049) = (2\,156\,201,\ 1\,107\,625)$. And $(1, 1)$ is itself the eigenvector $\mathbf{v_2}$, so $A^{10}(1, 1) = 3^{10}(1, 1) = (59\,049,\ 59\,049)$.

4. A rotation by $30°$ turns **every** nonzero vector by $30°$, so no vector stays on its own line, and there is no real eigenvector. Algebraically, trace $2\cos 30° = \sqrt{3}$ and determinant 1 give $\lambda^2 - \sqrt{3}\lambda + 1 = 0$, with discriminant $3 - 4 < 0$. The eigenvalues are $\cos 30° \pm i \sin 30° = e^{\pm i\pi/6}$: complex numbers of size 1 at angle $\pm 30°$, encoding "rotate by 30°, no scaling".

5. Steady state: $-0.2x + 0.3y = 0$, so $x = 1.5y$; with $x + y = 1$, $y = 0.4$, $x = 0.6$. Steady state $(0.6, 0.4)$. Trace 1.5, so the second eigenvalue is $1.5 - 1 = 0.5$ (check: determinant $0.56 - 0.06 = 0.5 = 1 \times 0.5$ ✓). The distance from equilibrium halves every step.

6. Write the start as a combination of eigenvectors; after k steps each component is multiplied by λ_i^k, and the dominant one outgrows the rest, so any start with a nonzero component along the dominant eigenvector converges to it (a random start almost surely has one). Convergence is slow when $|\lambda_2/\lambda_1|$ is close to 1, i.e. when the two largest eigenvalues are nearly equal in size; it fails if they're equal in size.

7. The covariance is already diagonal, so the principal components are the axes: PC1 = $\mathbf{e_1}$ (variance 4, sd 2), PC2 = $\mathbf{e_2}$ (variance 1, sd 1). PC1 explains $4/5 = 80\%$. The cloud is an axis-aligned ellipse, twice as wide (horizontally) as it is tall, with the two features uncorrelated.

8. V^T rotates (or reflects) the input so the special perpendicular input directions line up with the axes; Σ stretches each axis by a singular value (and may change dimension); U rotates the result into place. The matrix equals $\sum \sigma_i \mathbf{u}_i \mathbf{v}_i^\mathsf{T}$ with terms in decreasing importance; keeping the top k gives the best rank-k approximation, stored in $k(m + n + 1)$ numbers instead of mn, with error determined by the discarded singular values.

Part IX: Numerical Mathematics

Chapter 32. Numerical Mathematics: How Computers Actually Compute

Every earlier chapter quietly assumed that numbers are exact. Inside a computer they are not. This chapter shows how real numbers are actually stored, where the errors come from, how they grow or stay tame, and how to choose algorithms that give answers you can trust. We revisit root finding, derivatives, integrals, linear systems, differential equations and interpolation, this time as a numerical analyst sees them: understand the method, calculate a few steps by hand, implement it, and then experiment until it breaks.

Where this chapter sits

You need: scientific notation and powers of two (Volume 1, Chapter 3), Newton's method and Taylor-style approximations (Chapter 17, Chapter 18), the definite integral (Chapter 19), Gaussian elimination (Volume 1, Chapter 6 , Chapter 27), and limits and convergence (Chapter 16, Volume 1, Chapter 14). A little complex arithmetic (Volume 1, Chapter 15) helps for one aside.

You get: a working model of floating-point arithmetic; the vocabulary of error (absolute, relative, truncation, rounding, condition number, stability, order of convergence); a comparison of the standard numerical methods with guidance on choosing between them; and a checklist for deciding whether a computed number deserves belief.

It unlocks: the optimisation and machine-learning material of Volume 3, Chapter 33 , where every gradient step is a floating-point computation, and most of the practical work of scientific computing, simulation and data science.

32.1. The problem: the machine is not doing real-number arithmetic

Open a Python prompt and type:

```python
print(0.1 + 0.2)
print(0.1 + 0.2 == 0.3)
```

```
0.30000000000000004
False
```

This is not a Python bug. The same thing happens in C, Java, JavaScript, Excel and on your phone's calculator app if it is honest. The computer cannot store 0.1 exactly, it cannot store 0.2 exactly, and the sum of the two stored values is not the stored value of 0.3.

Most of the time the discrepancy is in the sixteenth significant digit and nobody cares. But errors can *grow*. Three real incidents give the flavour:

- In 1991 a Patriot missile battery in Dhahran failed to intercept an incoming Scud. Its clock counted tenths of a second in a 24-bit fixed-point register; 0.1 has no finite binary expansion, so every tick lost about 9.5×10^{-8} seconds. After 100 hours of continuous running the accumulated drift was about 0.34 s, enough for the target to be some 500 m from where the system looked.
- In 1996 the first Ariane 5 rocket self-destructed 37 seconds after launch when a 64-bit floating-point velocity value was converted to a 16-bit integer and overflowed.
- In 1982 the Vancouver Stock Exchange started a new index at 1000. It was recomputed thousands of times a day and *truncated* (chopped) to three decimal places each time rather than rounded. Twenty-two months later it stood near 525; recomputed correctly it should have been about 1099.

None of these was a failure of the mathematics on paper. Each was a failure to understand how the mathematics behaves once it is executed by a finite machine. That understanding is the subject of **numerical mathematics** (or **numerical analysis**): the study of algorithms that compute approximate answers to continuous problems, together with rigorous statements about how good those approximations are.



Numerical analysis

Numerical analysis is the branch of mathematics that designs and analyses algorithms for approximating answers to problems (roots, integrals, derivatives, solutions of equations) when exact symbolic answers are unavailable, too expensive, or meaningless because the data are themselves approximate. Its two central questions are *how fast does the method approach the answer?* and *how much error does it introduce?*

32.2. Floating point: how real numbers are stored

32.2.1. The problem

A real number such as π has infinitely many digits. Memory is finite. Something has to give. The question is *what* to give up so that the arithmetic is fast, predictable, and good enough for numbers as different as the mass of an electron (9.1×10^{-31} kg) and the number of atoms in the observable universe (about 10^{80}).

32.2.2. Intuition: scientific notation with a fixed budget of digits

You already know the answer from science class. Instead of writing 0.000000000000000000000000000000911, you write 9.11×10^{-31}: a handful of **significant digits** and an **exponent** that says where the decimal point goes. The decimal point "floats" to wherever it is needed; hence **floating point**.

A computer does the same in base 2, with a fixed budget: in the standard 64-bit "double precision" format, 53 significant binary digits (bits) and an 11-bit exponent. Fixing the number of significant digits rather than the number of decimal places has a crucial consequence: **the relative accuracy is**

the same everywhere, but the absolute spacing between representable numbers grows with their size. Between 1 and 2 there are 2^{52} evenly spaced doubles; between 1024 and 2048 there are also 2^{52}, spread over a range 1024 times wider.



Significand, exponent, floating point

A **floating-point number** has the form $\pm m \times \beta^{e}$, where β is the **base** (2 in hardware), m is the **significand** (also called the **mantissa**) holding a fixed number of significant digits, and e is the integer **exponent**. In base 2 the significand of a nonzero number can always be scaled so that it looks like $1.f_1 f_2 f_3 \ldots$ in binary; this is the **normalised** form, and the bits $f_1 f_2 \ldots$ are the **fraction**.

32.2.3. Why it works: the IEEE 754 double

Since 1985 essentially every processor has implemented the **IEEE 754** standard. A double occupies 64 bits split into three fields.

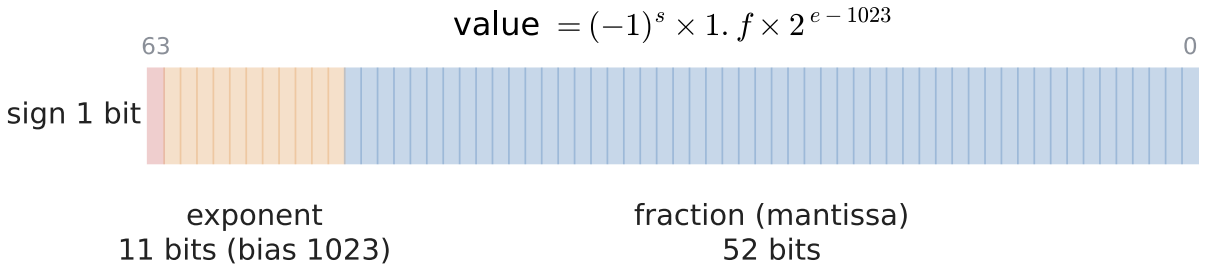


Figure 85. The 64 bits of an IEEE 754 double-precision number

For an ordinary ("normal") number the value is

$$x = (-1)^{s} \times \left(1 + \frac{f}{2^{52}}\right) \times 2^{\,E-1023},$$

where s is the sign bit, f is the 52-bit fraction read as an integer, and E is the 11-bit exponent field read as an integer between 1 and 2046. The leading 1 of the significand is not stored (it is always 1 for a normal number), which buys an extra bit for free: 52 stored bits give 53 bits of precision. The **bias** of 1023 lets the stored field be unsigned while the true exponent $E - 1023$ runs from -1022 to $+1023$.

The two leftover exponent patterns are reserved:

- $E = 2047$ (all ones) encodes **infinity** (fraction zero) and **NaN**, "not a number" (fraction nonzero), the result of operations such as $0/0$ or $\infty - \infty$.
- $E = 0$ (all zeros) encodes zero (there are two, $+0$ and -0) and the **subnormal** numbers, which drop the hidden leading 1 to fill the gap between zero and the smallest normal number.



Range, overflow, underflow

The largest finite double is about 1.80×10^{308}; the smallest positive normal double is about 2.23×10^{-308}; the smallest positive subnormal is about 4.9×10^{-324}. A result too large to represent **overflows** to $\pm\infty$. A result too small **underflows**, first gradually through the subnormals (losing precision) and finally to zero.

32.2.4. Worked example: what is actually stored for 0.1?

Let us convert 0.1 to binary by hand. To find binary digits after the point, repeatedly multiply by 2 and peel off the integer part:

$$
\begin{aligned}
0.1 \times 2 &= 0.2 \quad &&\to 0 \\
0.2 \times 2 &= 0.4 \quad &&\to 0 \\
0.4 \times 2 &= 0.8 \quad &&\to 0 \\
0.8 \times 2 &= 1.6 \quad &&\to 1 \quad \text{(keep 0.6)} \\
0.6 \times 2 &= 1.2 \quad &&\to 1 \quad \text{(keep 0.2)} \\
0.2 \times 2 &= 0.4 \quad &&\to 0 \quad \text{(we have seen 0.2 before: the pattern repeats)}
\end{aligned}
$$

So $0.1_{10} = 0.0\overline{0011}_2 = 0.000110011001100\ldots_2$. Like $1/3 = 0.333\ldots$ in decimal, it never terminates. The reason is that $0.1 = 1/10$ and 10 has the prime factor 5, which is not a factor of the base 2. Only fractions whose denominators are powers of 2 terminate in binary.

Normalise by moving the point four places right:

$$0.1 = 1.1001100110011\ldots_2 \times 2^{-4}.$$

- Sign: positive, so $s = 0$.
- Exponent: true exponent -4, so the stored field is $E = -4 + 1023 = 1019 = 01111111011_2$.
- Fraction: the 52 bits after the leading 1 are $1001\,1001\ldots1001$ followed by 1010; the infinite tail $1001\ldots$ beyond bit 52 starts with a 1, so the last kept group 1001 rounds up to 1010.

In hexadecimal the whole 64-bit word is `3FB999999999999A`; the final `A` (rather than `9`) is the round-up. Because we rounded *up*, the stored value is slightly *larger* than 0.1. Python can show its exact decimal value:

```python
from decimal import Decimal
import struct

print(Decimal(0.1))                    # exact value of the stored double
print(struct.pack('>d', 0.1).hex())    # its 64 raw bits in hex
```

```
0.1000000000000000055511151231257827021181583404541015625
3fb999999999999a
```

Now the puzzle of the opening dissolves. The doubles nearest to 0.1, 0.2 and 0.3 are

$$\begin{aligned}
\mathrm{fl}(0.1) &= 0.1000000000000000055511\ldots && \text{(a little high)}\\
\mathrm{fl}(0.2) &= 0.2000000000000000111022\ldots && \text{(a little high)}\\
\mathrm{fl}(0.3) &= 0.2999999999999999888977\ldots && \text{(a little low)}
\end{aligned}$$

Their exact sum is $0.3000000000000000166533\ldots$; rounding that to the nearest double gives $0.3000000000000000444089\ldots$, which is the double just *above* $\mathrm{fl}(0.3)$. Python prints the shortest decimal string that uniquely identifies it: `0.30000000000000004`.

32.2.5. Vocabulary: machine epsilon and the ulp



Machine epsilon and unit roundoff

Machine epsilon $\varepsilon_{\text{mach}}$ is the gap between 1 and the next larger floating-point number. For doubles $\varepsilon_{\text{mach}} = 2^{-52} \approx 2.22 \times 10^{-16}$. The **unit roundoff** $u = \varepsilon_{\text{mach}}/2 = 2^{-53} \approx 1.11 \times 10^{-16}$ is the largest relative error committed by rounding a real number (in range) to the nearest double. "Doubles carry about 16 significant decimal digits" is shorthand for $-\log_{10} u \approx 15.95$.



Ulp

An **ulp** (unit in the last place) of a floating-point number x is the gap between x and the next representable number of larger magnitude. $\mathrm{ulp}(1) = 2^{-52}$; $\mathrm{ulp}(10^{16}) = 2$. Errors are often quoted in ulps: "correctly rounded" means within half an ulp.

The single most useful model of floating-point arithmetic follows from these definitions. IEEE 754 requires that the basic operations $+, -, \times, \div$ and $\sqrt{\ }$ be **correctly rounded**: the result is the exact answer rounded to the nearest double. So for any of these operations, written $\circ$,

$$\mathrm{fl}(x \circ y) = (x \circ y)(1+\delta), \qquad |\delta| \le u.$$

Read it aloud: *each individual operation is wrong by a relative amount no larger than about* 10^{-16}. All the drama in this chapter comes not from a single operation but from how such tiny relative errors are amplified by what we do next.

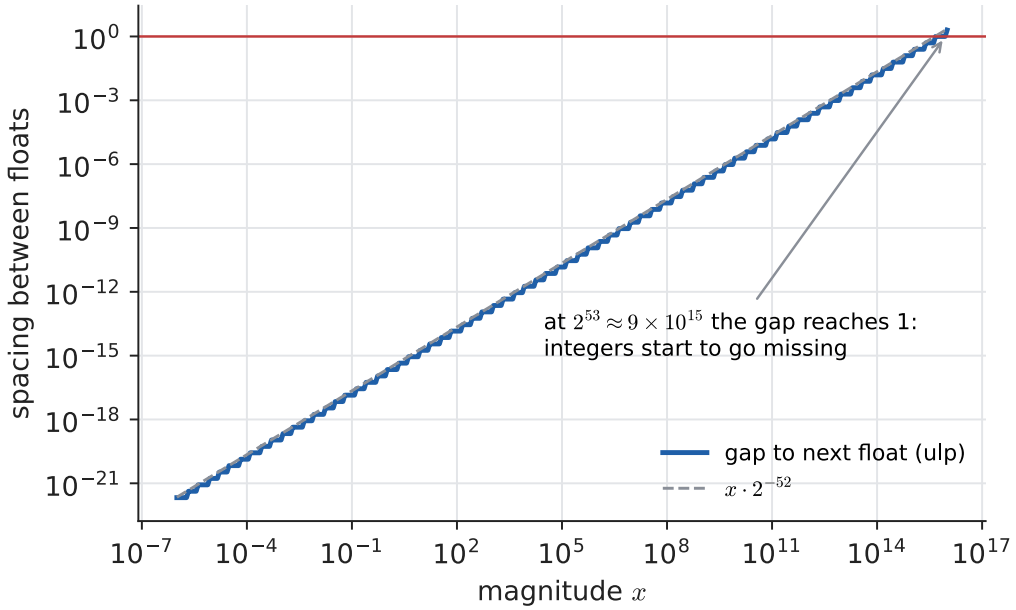


Figure 86. Gap between consecutive doubles grows in proportion to magnitude

The diagram shows the spacing between doubles as a function of their size. It is a staircase (constant within each power-of-two band, then doubling) that looks like a straight line on log-log axes. At $2^{53} = 9\,007\,199\,254\,740\,992$ the gap reaches 1, so $2^{53} + 1$ cannot be stored: `float(2**53 + 1)` returns `9007199254740992.0`. This is why JavaScript, whose only number type is the double, defines `Number.MAX_SAFE_INTEGER` as $2^{53} - 1$.



Never test floats for exact equality

`if x == 0.3:` is almost always a bug when `x` came out of arithmetic. Compare with a tolerance that combines a relative and an absolute part:

```python
import math
print(math.isclose(0.1 + 0.2, 0.3, rel_tol=1e-9, abs_tol=1e-12))   # True
```

A purely relative tolerance fails near zero (nothing is relatively close to 0 except 0); a purely absolute one fails for large numbers (10^{20} and $10^{20} + 1$ are "far apart" in absolute terms but identical as doubles). Note also that `NaN != NaN` by definition, so `x != x` is the classic NaN test.

Computer-science connection: choosing a number format

- **Money** should not be stored in binary floating point; use integer cents or a decimal type (`decimal.Decimal`, SQL `DECIMAL`), because decimal fractions such as 0.10 are exact there.
- **Machine learning** deliberately uses *less* precision: 32-bit floats (24-bit significand, about 7 digits), 16-bit half precision (11-bit significand, max value 65504), and Google's **bfloat16**, which keeps float32's 8-bit exponent (and so its range) but only an 8-bit significand. Neural networks tolerate the noise; what they cannot tolerate is overflow, which is why the exponent range was preserved.
- **Integers in JSON** larger than 2^{53} (tweet IDs, database keys) are silently corrupted by JavaScript parsers; APIs send them as strings for that reason.
- **Game engines** place the camera near the origin and move the world, because positions far from the origin have coarse ulps and objects visibly jitter ("floating-point jitter" at the edge of huge open-world maps).



The man who designed your arithmetic

Before 1985 every manufacturer had its own floating-point format, with its own rounding quirks; porting numerical code was a nightmare, and some machines could produce $x \neq y$ yet $x - y = 0$. William Kahan of Berkeley led the design of IEEE 754 (working closely with Intel on the 8087 coprocessor) and received the 1989 Turing Award for it. Subnormal numbers, his idea, guarantee that $x - y = 0$ if and only if $x = y$.



Float explorer

Type any decimal number and see its 64 bits coloured by field, its exact stored value, the neighbouring doubles on either side, and its ulp. Drag a slider through magnitudes from 10^{-320} to 10^{308} and watch the spacing change, the subnormal region appear, and the value finally overflow to infinity.

Try it: Float explorer: bits and ulp · The smallest subnormal double

32.2.6. Where you'd use this

- **Maps and GPS.** A latitude of 51.5° stored as a 32-bit float can only take values about 3.8×10^{-6} degrees apart, which on the ground is about 0.42 m. Fine for a running app; useless for a surveyor or a self-driving car, which store coordinates as doubles (gap about a nanometre). The same arithmetic explains why a flight simulator that stores the plane's position in metres from the centre of the Earth (about 6.4×10^{6} m) in float32 can only move it in 0.5 m jumps.
- **Money and spreadsheets.** A program that adds 0.1 ten times gets `0.9999999999999999`, so a test for equality with 1 fails. Accounting systems avoid this by counting in integer pennies; when you see a bank statement off by a penny, a binary-float rounding is a prime suspect.

- **Science:** astronomers and physicists who need more than 16 digits (pulsar timing, long-baseline interferometry) use double-double or quadruple-precision arithmetic, because the ulp of a double is too coarse for the quantities they measure.

US curriculum

Binary representation of numbers appears in **AP Computer Science Principles** (grades 10–12). Floating-point formats, rounding error and machine epsilon are not in the standard high-school math sequence; they belong to an undergraduate **Numerical Analysis** or **Scientific Computing** course (usually after Calculus II and Linear Algebra) and to **Computer Organization/Architecture** in a CS degree.

32.3. Rounding error and catastrophic cancellation

32.3.1. The problem

Solve $x^2 + 10^8 x + 1 = 0$. The quadratic formula from Volume 1, Chapter 8 gives *Try it:* Cancellation and the stable rewrite .

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-10^8 \pm \sqrt{10^{16} - 4}}{2}.$$

The two roots are about -10^8 and about -10^{-8} (their product is $c/a = 1$). In double precision:

```python
import math
a, b, c = 1.0, 1e8, 1.0
d = math.sqrt(b*b - 4*a*c)
print((-b + d) / (2*a))      # the small root, computed naively
print((-b - d) / (2*a))      # the large root
```

```
-7.450580596923828e-09
-100000000.0
```

The large root is fine. The small root should be $-1.0000000000000000\ldots \times 10^{-8}$; we got -7.45×10^{-9}, wrong in the *first* significant digit. A 25% error from a calculation in which every single operation was accurate to 16 digits. What happened?

32.3.2. Intuition: subtraction of nearly equal numbers



Absolute and relative error

If $\hat{x}$ approximates x, the **absolute error** is $|\hat{x} - x|$ and the **relative error** is $|\hat{x} - x|/|x|$ (for $x \neq 0$). Relative error 10^{-k} means roughly k correct **significant digits**. Floating point controls relative error; that is its whole design.

Here $b = 10^8$ and $\sqrt{b^2 - 4}$ are equal in their first 16 digits:

$$\sqrt{10^{16}-4} = 10^8\sqrt{1-4\times10^{-16}} \approx 10^8 - 2\times10^{-8}.$$

The computed square root carries a relative error up to u, i.e. an *absolute* error up to about $10^8 \times 10^{-16} = 10^{-8}$. Subtraction $-b+d$ is itself performed exactly (the two numbers are so close that the difference is representable), but it removes the 16 digits they share and leaves only what was in the "noise" region. The absolute error of about 10^{-8} is now attached to a result of size 2×10^{-8}: relative error of order 1. This is **catastrophic cancellation**.

The key insight: **subtraction does not create error; it exposes error that was already present in its inputs.** If the inputs were exact, the difference would be exact.

32.3.3. The method: rewrite to avoid the subtraction

The cure is algebra. Rationalise the numerator (multiply top and bottom by the conjugate):

$$\begin{aligned} x_{\text{small}} &= \frac{-b+\sqrt{b^2-4ac}}{2a}\cdot\frac{-b-\sqrt{b^2-4ac}}{-b-\sqrt{b^2-4ac}} \\ &= \frac{b^2-(b^2-4ac)}{2a\left(-b-\sqrt{b^2-4ac}\right)} && \text{(difference of squares)} \\ &= \frac{4ac}{2a\left(-b-\sqrt{b^2-4ac}\right)} \\ &= \frac{2c}{-b-\sqrt{b^2-4ac}}. && \text{(cancel } 2a\text{)} \end{aligned}$$

Now the denominator *adds* two numbers of the same sign: no cancellation. With our numbers, $2c/(-b-d) = 2/(-2\times10^8) = -1.0\times10^{-8}$ to full precision. Equivalently, use the product of roots: $x_{\text{small}} = c/(a\,x_{\text{large}})$.

Example 40. Procedure — Defusing catastrophic cancellation

Input: a formula to be evaluated in floating point.

Steps:

1. Look for subtractions $A - B$ (or additions of opposite signs) where A and B can be nearly equal for the inputs you care about.
2. Check whether A and B are themselves the result of rounding (a square root, a cosine, a measured value). If they are exact, the subtraction is harmless.
3. If they are inexact, rewrite algebraically so the cancellation happens symbolically, not numerically: multiply by a conjugate, use a trigonometric identity, use a Taylor series for small arguments, or use a library function designed for the job.
4. Test the rewritten formula against a high-precision reference (Python's `decimal` or `fractions`) at extreme inputs.

Output: an algebraically equivalent formula whose floating-point evaluation keeps its relative accuracy.

Some standard rewrites:

Dangerous form	Safe form	Why
$1 - \cos x$ for small x	$2\sin^2(x/2)$	identity; no subtraction
$\sqrt{x+1} - \sqrt{x}$ for large x	$1/(\sqrt{x+1} + \sqrt{x})$	conjugate
$\ln(1+x)$ for small x	`math.log1p(x)`	$1 + x$ would round away x
$e^x - 1$ for small x	`math.expm1(x)`	same reason
$\sqrt{x^2 + y^2}$	`math.hypot(x, y)`	x^2 may overflow

For $x = 10^{-8}$, `1 - math.cos(x)` returns exactly `0.0` (because $\cos(10^{-8}) = 1 - 5 \times 10^{-17}$ rounds to 1), while `2*math.sin(x/2)**2` returns `5.000000000000005e-17`, correct to 16 digits.

32.3.4. Overflow and underflow in practice

The other two floating-point hazards are about range, not precision.

Overflow. Computing $\sqrt{x^2 + y^2}$ for $x = 10^{200}$ overflows at x^2 even though the answer, about 10^{200}, is perfectly representable. `math.hypot` rescales internally.

Underflow. Multiply the probabilities of 1000 independent events, each 0.01, and the true answer 10^{-2000} underflows to 0. Every probabilistic model that multiplies many likelihoods therefore works with **logarithms**: $\log \prod p_i = \sum \log p_i$ turns a product that underflows into a sum that is perfectly well behaved (Volume 1, Chapter 12).

Computer-science connection: the log-sum-exp trick

Machine-learning classifiers compute the **softmax** $e^{z_i} / \sum_j e^{z_j}$ (Volume 3, Chapter 33). With scores like $z = 1000$, e^{1000} overflows to infinity and the result is `inf/inf = nan`. The fix uses the identity

$$\log \sum_j e^{z_j} = m + \log \sum_j e^{z_j - m}, \qquad m = \max_j z_j.$$

After subtracting the maximum, every exponent is at most $e^0 = 1$, so nothing overflows, and at least one term equals 1, so the sum cannot underflow to zero. Every deep-learning framework implements softmax and cross-entropy this way.



Cancellation in disguise

The textbook variance formula $\frac{1}{n}\sum x_i^2 - \bar{x}^{\,2}$ (Volume 3, Chapter 31) subtracts two large, nearly equal numbers when the data have a large mean and small spread (timestamps in nanoseconds, say). It can even return a *negative* variance. Use the two-pass formula $\frac{1}{n}\sum (x_i - \bar{x})^2$, or Welford's one-pass update, instead.

32.4. Conditioning versus stability

32.4.1. The problem

When a computed answer is bad, whose fault is it? Sometimes the *problem* is so sensitive that no algorithm could do better; sometimes a perfectly reasonable problem was solved with a *bad algorithm*. Numerical analysts keep these two ideas strictly separate, and you should too.

32.4.2. Conditioning: a property of the problem

Consider the system *Try it:* Nudge the right-hand side ; Feel $\kappa \approx 4 \times 10^4$.

$$\begin{aligned} x + y &= 2 \\ x + 1.0001\,y &= 2.0001 \end{aligned}$$

Subtracting the first equation from the second gives $0.0001\,y = 0.0001$, so $y = 1$ and $x = 1$. Now change the right-hand side of the second equation by one part in 20,000, from 2.0001 to 2.0002. Then $0.0001\,y = 0.0002$, so $y = 2$ and $x = 0$. A change in the fifth significant digit of the data moved the answer from $(1, 1)$ to $(0, 2)$.

Geometrically, the two equations are lines that are almost parallel (Volume 1, Chapter 6). Nudge either one and their intersection slides a long way along them. No algorithm, however careful, can fix that: if the data carry an uncertainty of 10^{-4}, the answer is uncertain in its first digit.



Condition number

The **condition number** κ of a problem measures how much relative changes in the input are amplified in the output:

$$\frac{\text{relative change in output}}{\text{relative change in input}} \approx \kappa.$$

For a differentiable function of one variable, $\kappa(x) = \left| \dfrac{x\,f'(x)}{f(x)} \right|$. For a linear system $A\mathbf{x} = \mathbf{b}$, $\kappa(A) = \|A\|\,\|A^{-1}\|$ (Chapter 27). A problem is **well-conditioned** if κ is modest and **ill-conditioned** if it is large. For the matrix above, $\kappa \approx 4 \times 10^{4}$.

Rule of thumb: **you can lose about $\log_{10}\kappa$ significant digits.** With 16 digits available and $\kappa = 4 \times 10^{4}$, expect about 11 or 12 good digits from any decent solver, if the data themselves are exact.

The function $f(x) = x - 1$ near $x = 1$ is a one-variable example: $\kappa = |x \cdot 1/(x-1)|$, which blows up as $x \to 1$. That is catastrophic cancellation seen through the lens of conditioning: subtracting nearly equal numbers is an ill-conditioned operation.

32.4.3. Stability: a property of the algorithm

An algorithm is **stable** if it does not add much error beyond what the conditioning of the problem forces. The gold standard is **backward stability**: the computed answer is the exact answer to a slightly perturbed problem (perturbed by a few multiples of u). Then

$$\text{forward error} \lesssim \kappa \times \text{backward error}.$$

This one line is the most important idea in numerical analysis. It splits every error into "the problem's sensitivity" times "the algorithm's sloppiness".

32.4.4. Worked example: one problem, a stable and an unstable algorithm

Define $I_n = \int_0^1 x^n e^{x-1}\,dx$ for $n = 0, 1, 2, \ldots$. Integration by parts (Chapter 19) with $u = x^n$, $dv = e^{x-1}dx$ gives a recurrence: *Try it:* Forwards versus backwards recurrence .

$$\begin{aligned} I_n &= \Big[x^n e^{x-1}\Big]_0^1 - \int_0^1 n x^{n-1} e^{x-1}\,dx \\ &= 1 - n\,I_{n-1}, \qquad I_0 = 1 - e^{-1}. \end{aligned}$$

Every I_n is positive (the integrand is positive) and they decrease (x^n shrinks on $[0,1]$). Run the recurrence forwards in Python:

```python
import math
I = 1 - 1/math.e                # I_0
for n in range(1, 21):
    I = 1 - n * I
    if n in (5, 10, 15, 17, 18, 19, 20):
        print(n, I)
```

```
5 0.14553294057308008
10 0.0838770700582927
15 0.059033793641901866
17 0.05719187059730757
18 -0.029453670751536265
19 1.559619744279189
20 -30.19239488558378
```

A negative value at $n = 18$, then nonsense. Why? I_0 is stored with error $\delta \approx 10^{-17}$. Each step multiplies the inherited error by $-n$, so after 20 steps it has been multiplied by $20! \approx 2.4 \times 10^{18}$. The algorithm is wildly **unstable**, even though the problem (compute I_{20}) is perfectly well-conditioned.

Now run the same recurrence **backwards**: $I_{n-1} = (1 - I_n)/n$. Each step *divides* the error by n. Start at $n = 40$ with the crude guess $I_{40} = 0$ (true value about 0.024, so the starting error is about 0.024):

```python
J = 0.0                         # crude guess for I_40
for n in range(40, 20, -1):
    J = (1 - J) / n             # gives I_{n-1}
print(J)                        # I_20
```

```
0.045544884075818054
```

The starting error has been divided by $40 \cdot 39 \cdots 21 \approx 3.4 \times 10^{29}$; the answer is correct to all 16 digits. Same mathematics, same recurrence, opposite direction, opposite fate.



Ill-conditioned or unstable?

If a result is bad, first ask whether the problem is ill-conditioned (then improve the data, reformulate the question, or accept the uncertainty) or the algorithm unstable (then change the algorithm). Running in higher precision helps both, but only postpones the problem in the unstable case: with the forward recurrence above, quadruple precision merely moves the collapse from $n = 18$ to about $n = 30$.

32.4.5. Where you'd use this

- **Navigation: GPS geometry.** A GPS receiver solves for its position from distances to several satellites. If the satellites in view are bunched together in one part of the sky, the equations are like the nearly parallel lines above: ill-conditioned, so a few metres of timing noise becomes tens of

metres of position error. Receivers report this amplification as the **dilution of precision** (DOP), a condition number with a friendlier name; surveyors schedule work for times when the satellites are well spread out.

- **Engineering and science:** weather forecasting is famously ill-conditioned (the "butterfly effect"): tiny errors in today's measurements grow until forecasts beyond about two weeks carry no skill, whatever the algorithm. Forecasters respond by running **ensembles** of slightly perturbed forecasts and reporting the spread, an honest answer to an ill-conditioned problem.
- **Computing:** before blaming a library for a bad least-squares fit, check the condition number of the data matrix (`numpy.linalg.cond`); near-duplicate input features (a price in pounds and the same price in dollars) make it huge.

US curriculum

Sensitivity of solutions to the data appears informally in **Algebra I/II** when students see nearly parallel lines. Condition numbers and stability are university material (**Numerical Analysis**, **Numerical Linear Algebra**).

32.5. Root finding compared

32.5.1. The problem

Solve $f(x) = 0$ when algebra gives up: $\cos x = x$, $x^5 - x - 1 = 0$, or "what interest rate makes this loan's payments equal its present value?" You met Newton's method in Chapter 18. Here we put it alongside its rivals and compare them as algorithms: what they need, how fast they converge, and how they fail.

To measure errors exactly we use a test problem whose answer we know: $f(x) = x^2 - 2$, root $r = \sqrt{2} = 1.41421356237309\ldots$.

32.5.2. Vocabulary: how fast is fast?



Order of convergence

Let $e_n = |x_n - r|$ be the error after n steps. A method **converges with order p** if

$$e_{n+1} \approx C\, e_n^{p}$$

for some constant $C > 0$ as $n \to \infty$. **Linear** convergence ($p = 1$, need $C < 1$) adds a fixed number of correct digits per step. **Quadratic** convergence ($p = 2$) roughly *doubles* the number of correct digits per step. **Superlinear** means anything faster than linear.

32.5.3. Bisection: slow, certain

Intuition. If a continuous function is negative at a and positive at b, it must cross zero in between (the Intermediate Value Theorem, Chapter 16). Check the midpoint; keep whichever half still has a sign change. It is binary search on the real line. *Try it:* Race all four methods .

Example 41. Procedure — Bisection

> **Input:** continuous f, an interval $[a, b]$ with $f(a)$ and $f(b)$ of opposite signs, a tolerance τ.
>
> **Steps:**
>
> 1. Compute the midpoint $m = (a + b)/2$ and $f(m)$.
> 2. If $f(a)$ and $f(m)$ have opposite signs (or $f(m) = 0$), set $b \leftarrow m$; otherwise set $a \leftarrow m$.
> 3. Repeat until $(b - a)/2 < \tau$.
>
> **Output:** the midpoint of the final interval, within τ of a root.

By hand on $[1, 2]$ ($f(1) = -1 < 0$, $f(2) = 2 > 0$):

step	a	b	m	$f(m) = m^2 - 2$
1	1	2	1.5	0.25 (positive, so $b \leftarrow 1.5$)
2	1	1.5	1.25	-0.4375 (negative, so $a \leftarrow 1.25$)
3	1.25	1.5	1.375	-0.109375 ($a \leftarrow 1.375$)
4	1.375	1.5	1.4375	0.06640625 ($b \leftarrow 1.4375$)
5	1.375	1.4375	1.40625	-0.0224609375 ($a \leftarrow 1.40625$)
6	1.40625	1.4375	1.421875	0.021728515625

Each step halves the bracket, so after n steps the error is at most $(b - a)/2^n$. Reaching 10^{-10} from a bracket of width 1 needs $n \geq \log_2(10^{10}) \approx 33.2$, i.e. 34 steps. Linear convergence with ratio $\frac{1}{2}$: about one decimal digit every 3.3 steps.

32.5.4. Newton's method: fast, demanding

Newton replaces the curve by its tangent line at x_n and jumps to where the tangent crosses zero:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

For $f(x) = x^2 - 2$ this is $x_{n+1} = x_n - \dfrac{x_n^2 - 2}{2x_n} = \dfrac{x_n}{2} + \dfrac{1}{x_n}$. From $x_0 = 1$:

$$
\begin{aligned}
x_1 &= \tfrac{1}{2} + 1 = 1.5 \\
x_2 &= 0.75 + \tfrac{1}{1.5} = 0.75 + 0.6\overline{6} = 1.41\overline{6} \\
x_3 &= 0.708\overline{3} + \tfrac{1}{1.41\overline{6}} = 0.708\overline{3} + 0.70588235\ldots = 1.4142156862\ldots \\
x_4 &= 1.4142135623746899\ldots
\end{aligned}
$$

The errors are $8.6 \times 10^{-2}, 2.5 \times 10^{-3}, 2.1 \times 10^{-6}, 1.6 \times 10^{-12}$: the exponent roughly doubles each time. That is quadratic convergence. Step 5 hits the correctly rounded value of $\sqrt{2}$. (Step 6 can bounce to the neighbouring double and back; at the level of the last bit, rounding decides, not Newton. Stopping tests must allow for that.)

Why quadratic? Expand f about the root with Taylor's theorem: the tangent line captures the first-order term exactly, so the leftover error comes from the second-order term, which is proportional to e_n^2. Precisely, $e_{n+1} \approx \left| \dfrac{f''(r)}{2f'(r)} \right| e_n^2$.

32.5.5. The secant method: Newton without the derivative

Often f' is unavailable (f is a simulation, or a black box). Replace the tangent by the **secant** through the last two points, i.e. approximate $f'(x_n)$ by $\dfrac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$: *Try it:* Step the secant method .

$$x_{n+1} = x_n - f(x_n)\,\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}.$$

From $x_0 = 1, x_1 = 2$ (so $f(x_0) = -1, f(x_1) = 2$):

$$
\begin{aligned}
x_2 &= 2 - 2 \cdot \frac{2 - 1}{2 - (-1)} = 2 - \tfrac{2}{3} = 1.\overline{3} \\
x_3 &= 1.\overline{3} - \left(-\tfrac{2}{9}\right) \cdot \frac{1.\overline{3} - 2}{-\frac{2}{9} - 2} = 1.\overline{3} + \tfrac{2}{9} \cdot \frac{-\frac{2}{3}}{-\frac{20}{9}} = 1.\overline{3} + \tfrac{1}{15} = 1.4 \\
x_4 &= 1.41463414\ldots, \quad x_5 = 1.41421143\ldots, \quad x_6 = 1.4142135620573\ldots
\end{aligned}
$$

Errors: $8.1 \times 10^{-2}, 1.4 \times 10^{-2}, 4.2 \times 10^{-4}, 2.1 \times 10^{-6}, 3.2 \times 10^{-10}$. Each exponent is about 1.6 times the previous one: the order is the golden ratio $p = \frac{1+\sqrt{5}}{2} \approx 1.618$. Slower per step than Newton, but each step costs one new function evaluation instead of a function plus a derivative, so per unit of work the secant method is often faster.

32.5.6. Fixed-point iteration: the general pattern

Rewrite $f(x) = 0$ as $x = g(x)$ and iterate $x_{n+1} = g(x_n)$. A solution of $x = g(x)$ is a **fixed point** of g. Newton's method is itself fixed-point iteration with $g(x) = x - f(x)/f'(x)$. *Try it:* Watch x ← 2/x oscillate ; Fixed point with cobweb .

Why it works. Near the fixed point r, $g(x_n) - g(r) \approx g'(r)(x_n - r)$, so $e_{n+1} \approx |g'(r)|\, e_n$. The iteration converges (for starting points close enough) when $|g'(r)| < 1$, linearly with ratio $|g'(r)|$, and the smaller $|g'(r)|$, the faster.

Two rearrangements of $x^2 = 2$:

- $g(x) = 2/x$. Then $g'(\sqrt{2}) = -2/2 = -1$: the borderline. From $x_0 = 1$ the iterates are $2, 1, 2, 1, \ldots$, oscillating forever.
- $g(x) = x - \tfrac{1}{3}(x^2 - 2)$. Then $g'(\sqrt{2}) = 1 - \tfrac{2}{3}\sqrt{2} \approx 0.0572$. From $x_0 = 1$: $1.\overline{3},\ 1.4074,\ 1.41381,\ 1.414190, \ldots$, each error about 0.057 times the last, so about 1.24 digits per step; 10^{-12} after 10 steps.

Fixed-point thinking reappears everywhere: Jacobi iteration below, PageRank (Chapter 28), value iteration in reinforcement learning, and the convergence proofs of many optimisers.

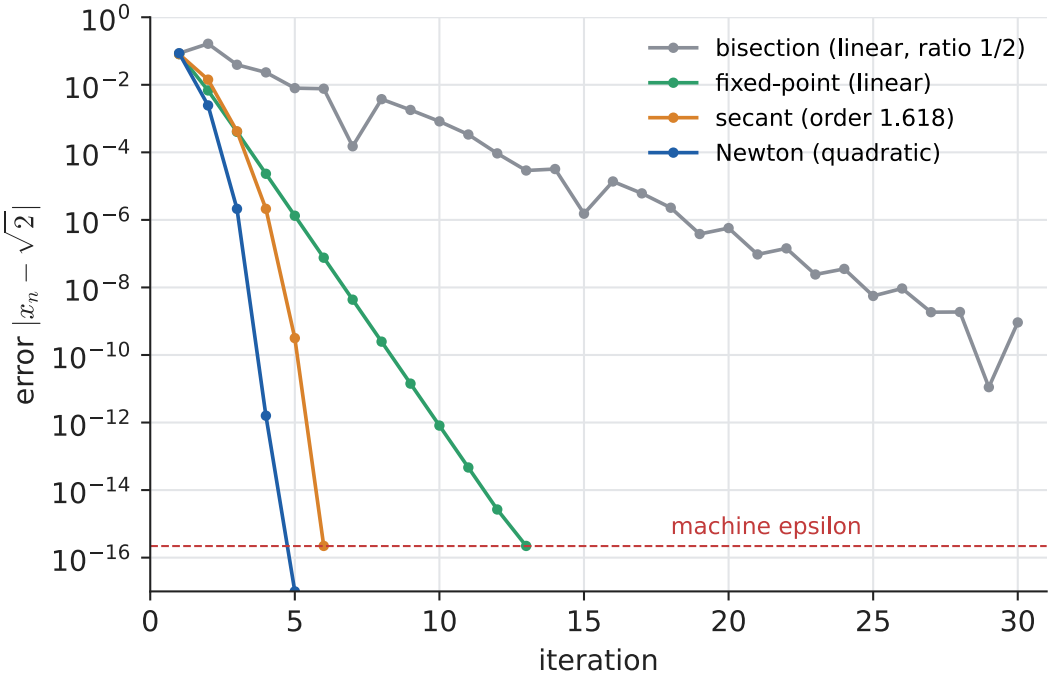


Figure 87. Root-finding methods compared on $x^2 - 2 = 0$

32.5.7. Choosing a method

Method	Needs	Order p	Guaranteed?	Typical failure
Bisection	sign-change bracket	1 (ratio $\frac{1}{2}$)	yes, if f continuous	slow; cannot find double roots like $x^2 = 0$ (no sign change)
Fixed-point	a rearrangement $x = g(x)$	1 (ratio $\lvert g'(r)\rvert$)	only if $\lvert g'\rvert < 1$ near root	diverges or oscillates when $\lvert g'(r)\rvert \geq 1$

Method	Needs	Order	Guaranteed?	Typical failure
Secant	two starting points	1.618	no	division by $f(x_n) - f(x_{n-1}) \approx 0$; can wander off
Newton	f and f', one start	2	no	$f' \approx 0$; cycles; jumps to a far root; only linear at multiple roots

Decision rule. If you can bracket the root and you only need it once, bisection's certainty is worth its slowness. If you need the root to full precision, or millions of times, use a fast method, ideally **safeguarded**: production libraries (for example `scipy.optimize.brentq`) use **Brent's method**, which tries secant-like and inverse-quadratic steps but falls back to bisection whenever a fast step would leave the bracket. You get guaranteed convergence and superlinear speed. Use Newton when derivatives are cheap and a good starting point is known (square roots in hardware, inside optimisers, implicit ODE solvers).

Here are all four, ready for experiments:

PYTHON

```python
import math

def bisection(f, a, b, tol=1e-12):
    fa = f(a)
    while (b - a) / 2 > tol:
        m = (a + b) / 2
        fm = f(m)
        if fa * fm <= 0:
            b = m
        else:
            a, fa = m, fm
    return (a + b) / 2

def newton(f, df, x, tol=1e-12, max_iter=50):
    for _ in range(max_iter):
        step = f(x) / df(x)
        x -= step
        if abs(step) < tol:
            return x
    raise RuntimeError("Newton did not converge")

def secant(f, x0, x1, tol=1e-12, max_iter=50):
    for _ in range(max_iter):
        f0, f1 = f(x0), f(x1)
        if f1 == f0:
            break                       # flat secant: cannot continue
        x0, x1 = x1, x1 - f1 * (x1 - x0) / (f1 - f0)
        if abs(x1 - x0) < tol:
            break
    return x1

def fixed_point(g, x, tol=1e-12, max_iter=200):
    for _ in range(max_iter):
        x_new = g(x)
        if abs(x_new - x) < tol:
            return x_new
        x = x_new
    raise RuntimeError("fixed-point iteration did not converge")

f = lambda x: x*x - 2
print(bisection(f, 1, 2))
print(newton(f, lambda x: 2*x, 1.0))
print(secant(f, 1.0, 2.0))
print(fixed_point(lambda x: x - (x*x - 2)/3, 1.0))
print(math.sqrt(2))
```

```
1.4142135623724243
1.414213562373095
1.4142135623730951
1.4142135623730485
1.4142135623730951
```

Bisection and fixed-point iteration stop once their steps fall below 10^{-12}, so they agree with $\sqrt{2}$ to about 12 digits, exactly as their tolerances promise. Secant reaches the correctly rounded value; Newton finishes one ulp below it (the last-bit bounce mentioned earlier). Each method delivers what

its stopping test asked for, and no more.



Stopping criteria

"Stop when $|f(x_n)| < \tau$" is fragile: if f is very flat near the root, $|f|$ is tiny far from it; if it is very steep, $|f|$ may never get below τ because of rounding. "Stop when $|x_{n+1} - x_n| < \tau$" is better but should use a relative tolerance for large roots (an absolute tolerance of 10^{-12} is below the ulp of 10^{6}). Always cap the number of iterations.



Newton's basins

Plot $f(x) = x^3 - 2x + 2$ and drag the starting point for Newton's method. From $x_0 = 0$ the iterates cycle $0 \to 1 \to 0 \to \ldots$ forever. Colour each starting point by which root (if any) it reaches, then switch to the complex plane (Volume 1, Chapter 15) for $z^3 - 1 = 0$ and watch the famous fractal Newton basins appear.

Try it: Complex basins of $z^3 - 1$ · Cycle islands of $z^3 - 2z + 2$ · Open Newton's basins

Computer-science connection: Newton inside your CPU

Hardware and math libraries compute $1/\sqrt{x}$ and division by a table lookup for a few correct bits followed by one or two Newton steps, each doubling the correct bits (8 to 16 to 32 ...). The notorious "fast inverse square root" in the Quake III source code (1999) is exactly this: a bit-level trick on the IEEE 754 representation to get a rough first guess, then a single Newton step.

32.5.8. Where you'd use this

- **Everyday finance: what is this loan really costing?** Borrow £10,000 and repay £880 a month for 12 months: £10,560 in all, which a salesperson might call a "5.6% flat rate". The true monthly rate r solves $880\,\frac{1-(1+r)^{-12}}{r} = 10\,000$, which has no algebraic solution. Bisection or Newton gives $r \approx 0.848\%$ a month, an effective annual rate of $1.00848^{12} - 1 \approx 10.7\%$, nearly double the flat rate. Spreadsheet functions `RATE` and `IRR` are root finders.
- **Physics: Kepler's equation.** To find where a planet or satellite is at time t, you must solve $M = E - e\sin E$ for the angle E, given the mean anomaly M and the eccentricity e. Every orbit-prediction program (including the one in your phone's GPS chip, which needs satellite positions) does this with a few Newton steps.
- **Engineering:** the Colebrook equation for friction in a pipe, the equilibrium of a chemical reaction, and the operating point of a diode circuit are all "solve $f(x) = 0$" problems handed to a root finder.

US curriculum

The Intermediate Value Theorem is in **AP Calculus AB**; Newton's method appears in many **Calculus I** courses (as an application of derivatives). Bisection, secant and convergence orders are typically first met in a university **Numerical Analysis** course.

32.6. Numerical differentiation: the V-shaped error

32.6.1. The problem

You have a function as code, say a physics simulation or a pricing model, and need its derivative. The definition (Chapter 17) suggests taking a small h:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

Mathematics says "the smaller h the better". The computer disagrees.

32.6.2. Intuition: two errors pulling in opposite directions

Truncation error comes from using a finite h instead of a limit. Taylor's theorem (the polynomial approximation of Chapter 18) says

$$f(x+h) = f(x) + hf'(x) + \tfrac{h^2}{2} f''(\xi) \quad \text{for some } \xi \text{ between } x \text{ and } x+h,$$

so

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \underbrace{\tfrac{h}{2} f''(\xi)}_{\text{truncation}}.$$

Truncation error is proportional to h: it shrinks as h shrinks.

Rounding error comes from the subtraction $f(x+h) - f(x)$: two nearly equal numbers, each carrying a rounding error of about $u|f(x)|$. That is catastrophic cancellation again. The absolute error of the numerator is about $u|f(x)|$, and we then divide by h, giving an error of about $u|f(x)|/h$, which *grows* as h shrinks.

32.6.3. Why it works: finding the best h

Total error is approximately

$$E(h) \approx \frac{M}{2} h + \frac{u\,|f|}{h}, \qquad M \approx |f''|.$$

Minimise with calculus: $E'(h) = \frac{M}{2} - \frac{u|f|}{h^2} = 0$ gives $h^* = \sqrt{2u|f|/M}$, which is of order $\sqrt{u} \approx 10^{-8}$ when f and f'' are of order 1. At that h both terms are about 10^{-8}: **the forward difference can only ever deliver about half the available digits.**

The **central difference** is smarter. Subtract the Taylor expansions of $f(x+h)$ and $f(x-h)$; the f'' terms cancel:

$$
\begin{aligned}
f(x+h) &= f(x) + hf'(x) + \tfrac{h^2}{2}f''(x) + \tfrac{h^3}{6}f'''(x) + \cdots \\
f(x-h) &= f(x) - hf'(x) + \tfrac{h^2}{2}f''(x) - \tfrac{h^3}{6}f'''(x) + \cdots \\
f(x+h) - f(x-h) &= 2hf'(x) + \tfrac{h^3}{3}f'''(x) + \cdots \\
\frac{f(x+h) - f(x-h)}{2h} &= f'(x) + \tfrac{h^2}{6}f'''(x) + \cdots
\end{aligned}
$$

Truncation error is now $O(h^2)$. Balancing h^2 against u/h gives $h^* \sim u^{1/3} \approx 6 \times 10^{-6}$ and a best error around $u^{2/3} \approx 10^{-11}$: about two-thirds of the digits.



Big-O notation for errors

$E(h) = O(h^p)$ as $h \to 0$ means $|E(h)| \le Ch^p$ for some constant C and all small enough h. A method with error $O(h^p)$ has **order** p: halving h divides the error by about 2^p. This is the same symbol as in algorithm analysis, but there $n \to \infty$; here $h \to 0$.

32.6.4. Worked example

Take $f(x) = \sin x$ at $x = 1$, where the true derivative is $\cos 1 = 0.5403023058681398$. *Try it:* Plot the V-shaped error .

h	forward	forward error	central	central error
10^{-1}	0.4973637525	4.3×10^{-2}	0.5394022522	9.0×10^{-4}
10^{-2}	0.5360859810	4.2×10^{-3}	0.5402933009	9.0×10^{-6}
10^{-4}	0.5402602314	4.2×10^{-5}	0.5403023050	9.0×10^{-10}
10^{-6}	0.5403018851	4.2×10^{-7}	0.5403023059	2.8×10^{-11}
10^{-8}	0.5403022918	1.4×10^{-8}	0.5403023029	3.0×10^{-9}
10^{-10}	0.5403022474	5.8×10^{-8}	0.5403022474	5.8×10^{-8}
10^{-12}	0.5403455461	4.3×10^{-5}	0.5402900349	1.2×10^{-5}

Read the forward column top to bottom: the error falls by a factor of 10 each time h does (order 1), bottoms out near $h = 10^{-8}$, then *rises* again. The central column falls by a factor of 100 per decade of h (order 2), bottoms out near $h = 10^{-6}$, then rises. Plot the error against h on log–log axes and you get the characteristic V.

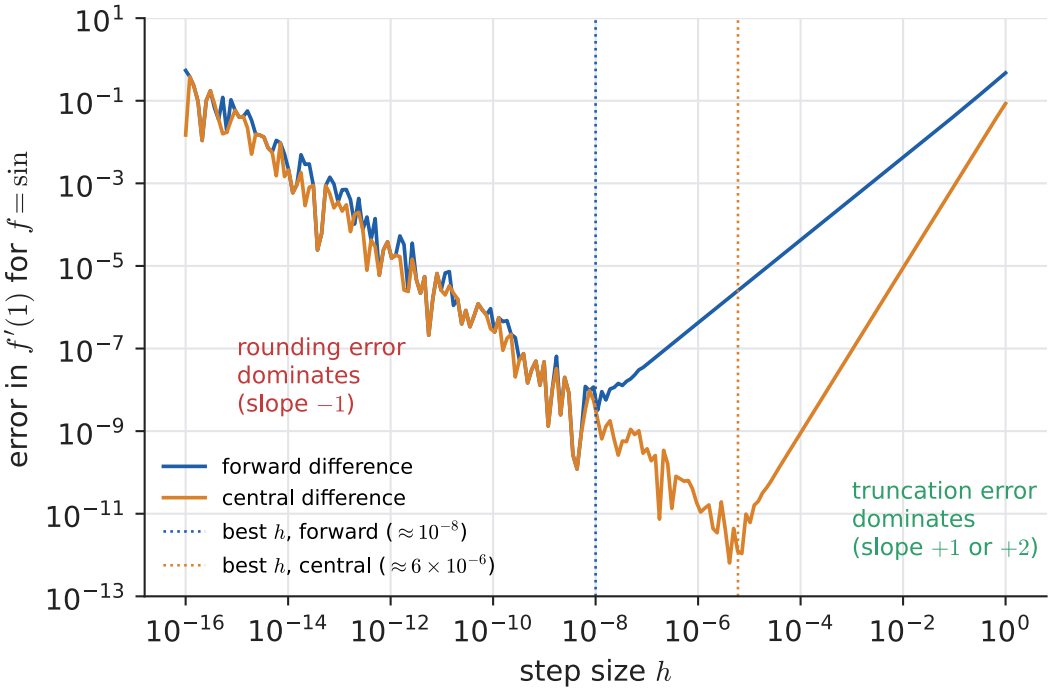


Figure 88. Error of finite-difference derivatives: truncation on the right, rounding on the left

32.6.5. Alternative approaches

- **Complex-step differentiation.** If f is built from operations that make sense for complex numbers (Volume 1, Chapter 15), then $f(x + ih) \approx f(x) + ihf'(x)$, so $f'(x) \approx \operatorname{Im} f(x + ih)/h$. No subtraction occurs, so there is no cancellation and h can be absurdly small. With $h = 10^{-20}$, `cmath.sin(1 + 1e-20j).imag / 1e-20` returns `0.5403023058681398`, correct to every digit. *Try it:* Try the complex step .
- **Automatic differentiation** (next sidebar) gives exact derivatives of code, to rounding error, with no h at all.
- **Symbolic differentiation** (a computer algebra system) gives a formula, which is ideal when f is a formula, impossible when it is a 10,000-line simulation.

Computer-science connection: automatic differentiation

Training a neural network needs the gradient of a loss with respect to millions of parameters. Finite differences would need millions of extra function evaluations, each with only 8 good digits. Instead, frameworks such as PyTorch and JAX use **automatic differentiation**: they record the elementary operations the program performs and apply the chain rule (Chapter 17, Chapter 20) to each, exactly. Reverse-mode AD ("backpropagation") delivers the full gradient for a small constant multiple of the cost of one function evaluation. Finite differences survive as the standard way to *test* an AD implementation: a "gradient check".



Halving h

A slider sets $\log_{10} h$ from 0 to -16. Show the secant line on the graph of f, the computed slope, and a live point moving along the V-shaped error plot. Switch between forward, central and complex-step formulas and switch the precision between float16, float32 and float64 to see the bottom of the V move.

Try it: Halving h in three precisions

32.6.6. Where you'd use this

- **Sport and fitness trackers.** A GPS watch records your position once a second and estimates your speed by a difference quotient. Position errors of a few metres divided by a 1-second step give speed errors of several metres per second, so the raw pace jumps about; watches smooth over several seconds, a larger h, trading a little truncation error for much less noise. That is the V-shaped trade-off, with measurement noise in place of rounding.
- **Engineering:** acceleration from measured velocity, flow rate from tank-level readings and the slope of a road from survey heights are all finite differences of noisy data; the larger the noise, the larger the best h.

US curriculum

Difference quotients are central to **AP Calculus AB** (estimating f' from tables of values is a standard exam question). The rounding-versus-truncation trade-off is university **Numerical Analysis**.

32.7. Numerical integration

32.7.1. The problem

Many integrals have no elementary antiderivative: $\int e^{-x^2}\,dx$ (the normal distribution, Volume 3, Chapter 30), $\int \sqrt{1+\cos^2 x}\,dx$ (arc length of a sine wave), and every integral of measured data. The Riemann sums of Chapter 19 converge, but slowly. Can we do better with the same number of function evaluations?

As a test integral with a known answer we use

$$\int_0^1 e^x\,dx = e - 1 = 1.718281828459045\ldots$$

32.7.2. Intuition and derivation

Split $[a, b]$ into n strips of width $h = (b-a)/n$, with nodes $x_i = a + ih$.

Trapezoid rule. Replace the curve on each strip by the straight line joining its endpoints. Each strip is a trapezoid with area $\frac{h}{2}\left(f(x_i) + f(x_{i+1})\right)$. Adding them, every interior node is counted twice:

$$T_n = h\left[\tfrac{1}{2}f(x_0) + f(x_1) + \cdots + f(x_{n-1}) + \tfrac{1}{2}f(x_n)\right].$$

The error is $-\dfrac{(b-a)h^2}{12}f''(\xi)$ for some ξ: order 2. (The sign tells you something too: for a convex function, $f'' > 0$, the chords lie above the curve and the trapezoid rule overestimates.)

Simpson's rule. Replace the curve on each *pair* of strips by the parabola through three points. Integrating that parabola (an exercise in Chapter 19-style algebra) gives weights $\frac{h}{3}(1, 4, 1)$ on each pair, and adding pairs (n even):

$$S_n = \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 4f(x_{n-1}) + f(x_n)\right].$$

The error is $-\dfrac{(b-a)h^4}{180}f^{(4)}(\xi)$: order 4. Surprisingly, it is exact for cubics, not just quadratics, because the cubic error term is odd about each pair's midpoint and cancels.

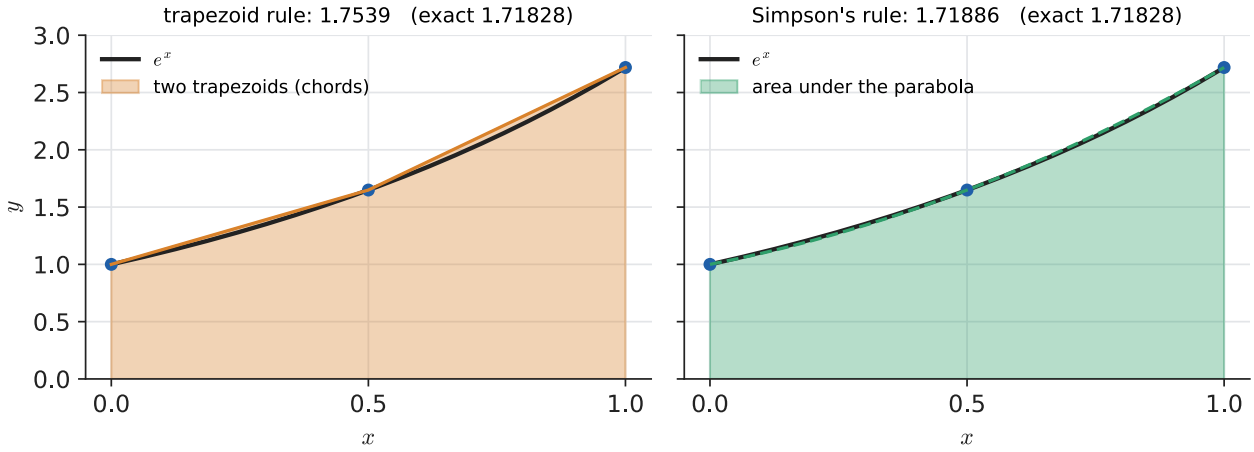


Figure 89. The two rules on $\int_0^1 e^x\,dx$ with $n = 2$. Left: the trapezoid rule replaces the curve by chords, which lie above a convex curve, so it overestimates. Right: Simpson's rule fits one parabola through the three points; it is indistinguishable from the curve at this scale.

32.7.3. Worked example: $n = 2$ by hand

With $n = 2$, $h = 0.5$, nodes $0, 0.5, 1$:

$$f(0) = 1, \qquad f(0.5) = e^{0.5} = 1.6487212707, \qquad f(1) = e = 2.7182818285.$$

Trapezoid:

$$\begin{aligned} T_2 &= 0.5\left[\tfrac{1}{2}(1) + 1.6487212707 + \tfrac{1}{2}(2.7182818285)\right] \\ &= 0.5\left[0.5 + 1.6487212707 + 1.3591409142\right] \\ &= 0.5 \times 3.5078621849 = 1.7539310925. \end{aligned}$$

Error $1.7539310925 - 1.7182818285 = 0.0356$, an overestimate, as predicted for convex e^x.

Simpson:

$$
\begin{aligned}
S_2 &= \frac{0.5}{3}[1 + 4(1.6487212707) + 2.7182818285] \\
&= \tfrac{1}{6}[1 + 6.5948850828 + 2.7182818285] \\
&= \tfrac{1}{6} \times 10.3131669113 = 1.7188611519.
\end{aligned}
$$

Error 5.8×10^{-4}, sixty times smaller with the same three function values.

Doubling n repeatedly confirms the orders:

n	T_n	error	S_n	error
2	1.7539310925	3.6×10^{-2}	1.7188611519	5.8×10^{-4}
4	1.7272219046	8.9×10^{-3}	1.7183188419	3.7×10^{-5}
8	1.7205185922	2.2×10^{-3}	1.7182841547	2.3×10^{-6}
16	1.7188411286	5.6×10^{-4}	1.7182819741	1.5×10^{-7}

Trapezoid errors drop by a factor of 4 ($= 2^2$) per doubling; Simpson errors by 16 ($= 2^4$).

32.7.4. Alternative: Monte Carlo integration

A completely different idea from probability (Volume 3, Chapter 29): the average value of f on $[a, b]$ is $\frac{1}{b-a}\int_a^b f$, so pick n uniformly random points and average:

$$
\int_a^b f(x)\,dx \approx (b-a) \cdot \frac{1}{n}\sum_{i=1}^{n} f(U_i), \qquad U_i \text{ uniform on } [a, b].
$$

By the central limit theorem (Volume 3, Chapter 30) the error is random with standard deviation $\sigma/\sqrt{n}$, where σ is the standard deviation of $f(U)$. To gain one decimal digit you need 100 times as many points. Dreadful in one dimension, as the plot shows. But the $1/\sqrt{n}$ rate **does not depend on the dimension**, whereas a grid-based rule in d dimensions with m points per axis needs m^d evaluations. For a 10-dimensional integral (pricing a basket option, rendering a pixel with light bouncing several times), Monte Carlo wins easily.

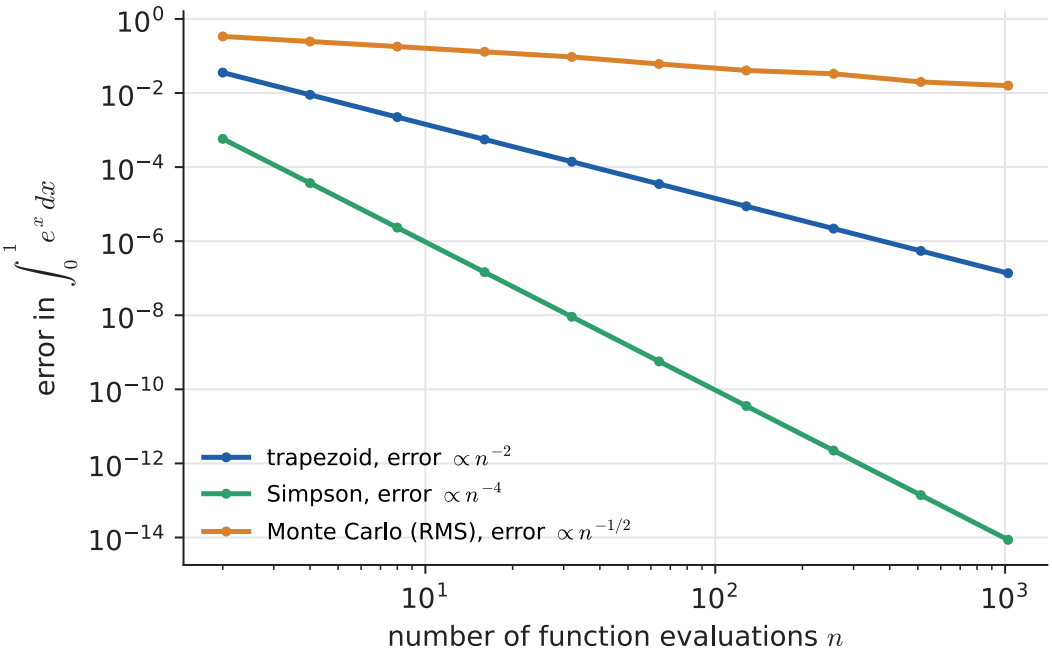


Figure 90. Error against work for three integration rules

PYTHON

```
import math, random

def trapezoid(f, a, b, n):
    h = (b - a) / n
    return h * (f(a)/2 + sum(f(a + i*h) for i in range(1, n)) + f(b)/2)

def simpson(f, a, b, n):
    if n % 2:
        raise ValueError("Simpson's rule needs an even number of strips")
    h = (b - a) / n
    odd  = sum(f(a + i*h) for i in range(1, n, 2))
    even = sum(f(a + i*h) for i in range(2, n, 2))
    return h / 3 * (f(a) + 4*odd + 2*even + f(b))

def monte_carlo(f, a, b, n, rng=random.Random(42)):
    return (b - a) * sum(f(rng.uniform(a, b)) for _ in range(n)) / n

exact = math.e - 1
for n in (4, 16, 64):
    print(n, trapezoid(math.exp, 0, 1, n) - exact,
             simpson(math.exp, 0, 1, n) - exact,
             monte_carlo(math.exp, 0, 1, n) - exact)
```

```
4 0.008940076098471472 3.7013462701906974e-05 -0.34643844509874033
16 0.000559300120949402 1.4559284666759709e-07 -0.14119251170139946
64 3.495839104905585e-05 5.689697601951593e-10 0.0009426857183749693
```

32.7.5. Choosing a method

- Smooth function, low dimension, can evaluate anywhere: Simpson, or better, an adaptive or Gaussian quadrature routine (`scipy.integrate.quad`), which places points cleverly and estimates its own error.

- Data at fixed, equally spaced points (sensor readings): trapezoid or Simpson on the data you have.
- Function with a kink or discontinuity: split the interval at the kink; higher-order rules lose their advantage across it.
- Many dimensions, or a complicated region: Monte Carlo (or quasi-Monte Carlo with low-discrepancy sequences, which approaches $1/n$).



Higher order is not always better

Simpson's error bound involves $f^{(4)}$. If f is not smooth ($\sqrt{x}$ near 0, $|x|$, a step), the fourth derivative is huge or undefined and Simpson's advantage evaporates; on $\int_0^1 \sqrt{x}\,dx$ both rules converge at only about order 1.5. Also, an error *estimate* is not an error *guarantee*: a function with a narrow spike between your sample points can fool any rule that only samples it.

Computer-science connection: Monte Carlo rendering

Every pixel in a path-traced film frame (Pixar has rendered this way since about 2013) is a high-dimensional integral over light paths. Renderers estimate it with Monte Carlo sampling; the "noise" in a quick preview render is exactly the $\sigma/\sqrt{n}$ error, which is why halving the noise costs four times the samples and why modern pipelines add machine-learned denoisers.

32.7.6. Where you'd use this

- **Medicine: drug exposure.** Blood concentration of a drug measured at 0, 1, 2, 4, 6 and 8 hours is 0, 8, 10, 7, 4, 2 mg/L. The total exposure, the **area under the curve** (AUC), is estimated by the trapezoid rule on the unequal intervals: $4 + 9 + 17 + 11 + 6 = 47$ mg·h/L. Regulators compare generic and brand-name drugs by exactly this number.
- **Everyday: area of an irregular garden or field.** Measure the width every 2 m along its length and apply the trapezoid or Simpson rule: the method surveyors and builders use for lawns, ponds and cut-and-fill earthworks.
- **Physics and engineering:** the energy used by a device is the integral of the power readings from a smart meter; the distance travelled by a car is the integral of its speedometer log; the impulse of a collision is the integral of a sampled force.

US curriculum

Left, right and midpoint Riemann sums and the **trapezoid rule** are explicitly in **AP Calculus AB**. Simpson's rule appears in many **Calculus II** textbooks. Monte Carlo methods appear in **AP Statistics** simulations and in university probability and numerical courses.

32.8. Solving linear systems on a computer

32.8.1. The problem

Systems $A\mathbf{x} = \mathbf{b}$ with thousands or millions of unknowns arise in structural engineering, circuit simulation, least-squares fitting (Volume 3, Chapter 33), and the discretisation of every partial differential equation. You know Gaussian elimination from Volume 1, Chapter 6 and Chapter 27. What changes when it runs in floating point?

32.8.2. Why pivoting is essential

Solve *Try it:* Eliminate with a 10^{-20} pivot .

$$\begin{aligned} 10^{-20}x + y &= 1 \\ x + y &= 2 \end{aligned}$$

The exact solution is extremely close to $x = 1, y = 1$. Naive elimination uses the tiny 10^{-20} as the **pivot**: multiply row 1 by $m = 1/10^{-20} = 10^{20}$ and subtract from row 2:

$$\begin{aligned} &(1 - 10^{20})\,y = 2 - 10^{20} && \text{(row 2 minus } 10^{20} \times \text{row 1)} \\ &\mathrm{fl}(1 - 10^{20}) = -10^{20}, \quad \mathrm{fl}(2 - 10^{20}) = -10^{20} && \text{(1 and 2 are far below one ulp of } 10^{20}) \\ &y = 1 \\ &x = \frac{1 - y}{10^{-20}} = \frac{0}{10^{-20}} = 0. \end{aligned}$$

$x = 0$ is completely wrong. The huge multiplier swamped row 2's information (the "1" and "2" were rounded away), and the back substitution then divided a cancellation error by a tiny number.

Partial pivoting swaps the rows first so that the pivot is the largest entry (in absolute value) in its column:

$$\begin{aligned} x + y &= 2 \\ 10^{-20}x + y &= 1 \end{aligned} \quad \Longrightarrow \quad (1 - 10^{-20})\,y = 1 - 2 \times 10^{-20} \quad \Longrightarrow \quad y = 1,\ x = 2 - y = 1.$$

Now the multiplier is 10^{-20}, at most 1 in size, and nothing gets swamped. With partial pivoting every multiplier satisfies $|m| \le 1$, which keeps the growth of rounding errors under control; in practice Gaussian elimination with partial pivoting is backward stable.



Pivot

In Gaussian elimination the **pivot** is the entry used to eliminate the entries below it in the same column. **Partial pivoting** chooses, at each column, the row with the largest absolute value in that column (among the rows not yet used) and swaps it into the pivot position. **Back substitution** solves the resulting upper-triangular system from the last equation upward.

Example 42. Procedure — Gaussian elimination with partial pivoting

Input: an $n \times n$ matrix A and vector $\mathbf{b}$ (as the augmented matrix $[A \mid \mathbf{b}]$).

Steps:

1. For each column $k = 1, \dots, n-1$:
   a. Find the row $p \geq k$ with the largest $|a_{pk}|$. If it is zero, the matrix is singular; stop.
   b. Swap rows p and k.
   c. For each row $i > k$: compute the multiplier $m_{ik} = a_{ik}/a_{kk}$ and replace row i by row i minus m_{ik} times row k.
2. Back substitution: for $i = n, n-1, \dots, 1$, set $x_i = \left(b_i - \sum_{j>i} a_{ij} x_j\right)/a_{ii}$.

Output: the solution $\mathbf{x}$. Cost: about $\tfrac{2}{3}n^3$ floating-point operations.

32.8.3. Worked example

Solve

$$\begin{aligned} 2x + y - z &= 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= -3 \end{aligned} \qquad \left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right].$$

Column 1. The entries are $2, -3, -2$; the largest in absolute value is -3 (row 2). Swap rows 1 and 2:

$$\left[\begin{array}{ccc|c} -3 & -1 & 2 & -11 \\ 2 & 1 & -1 & 8 \\ -2 & 1 & 2 & -3 \end{array}\right].$$

Multipliers: $m_{21} = 2/(-3) = -\tfrac{2}{3}$, $m_{31} = -2/(-3) = \tfrac{2}{3}$.

$$\begin{aligned} R_2 - (-\tfrac{2}{3})R_1 &= (2, 1, -1, 8) + \tfrac{2}{3}(-3, -1, 2, -11) = (0,\ \tfrac{1}{3},\ \tfrac{1}{3},\ \tfrac{2}{3}) \\ R_3 - \tfrac{2}{3}R_1 &= (-2, 1, 2, -3) - \tfrac{2}{3}(-3, -1, 2, -11) = (0,\ \tfrac{5}{3},\ \tfrac{2}{3},\ \tfrac{13}{3}) \end{aligned}$$

Column 2. Among rows 2 and 3 the entries are $\tfrac{1}{3}$ and $\tfrac{5}{3}$; pivot on $\tfrac{5}{3}$, so swap rows 2 and 3. Multiplier $m_{32} = \tfrac{1}{3}/\tfrac{5}{3} = \tfrac{1}{5}$:

$$(0, \tfrac{1}{3}, \tfrac{1}{3}, \tfrac{2}{3}) - \tfrac{1}{5}(0, \tfrac{5}{3}, \tfrac{2}{3}, \tfrac{13}{3}) = (0,\ 0,\ \tfrac{1}{3} - \tfrac{2}{15},\ \tfrac{2}{3} - \tfrac{13}{15}) = (0, 0, \tfrac{1}{5}, -\tfrac{1}{5}).$$

The triangular system is

$$\left[\begin{array}{ccc|c} -3 & -1 & 2 & -11 \\ 0 & \tfrac{5}{3} & \tfrac{2}{3} & \tfrac{13}{3} \\ 0 & 0 & \tfrac{1}{5} & -\tfrac{1}{5} \end{array}\right].$$

Back substitution.

$$\begin{aligned} \tfrac{1}{5}z = -\tfrac{1}{5} &\quad\Rightarrow\ z = -1 \\ \tfrac{5}{3}y + \tfrac{2}{3}(-1) = \tfrac{13}{3} &\quad\Rightarrow\ \tfrac{5}{3}y = \tfrac{15}{3} \ \Rightarrow\ y = 3 \\ -3x - 3 + 2(-1) = -11 &\quad\Rightarrow\ -3x = -6 \ \Rightarrow\ x = 2. \end{aligned}$$

Check in the original first equation: $2(2) + 3 - (-1) = 8$. Correct.

PYTHON

```python
import numpy as np

def solve_pp(A, b):
    """Gaussian elimination with partial pivoting, then back substitution."""
    A = np.array(A, dtype=float)
    b = np.array(b, dtype=float)
    n = len(b)
    for k in range(n - 1):
        p = k + np.argmax(np.abs(A[k:, k]))       # row of largest pivot
        if A[p, k] == 0:
            raise ValueError("matrix is singular")
        A[[k, p]], b[[k, p]] = A[[p, k]], b[[p, k]]  # swap rows k and p
        for i in range(k + 1, n):
            m = A[i, k] / A[k, k]
            A[i, k:] -= m * A[k, k:]
            b[i] -= m * b[k]
    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        x[i] = (b[i] - A[i, i+1:] @ x[i+1:]) / A[i, i]
    return x

print(solve_pp([[2, 1, -1], [-3, -1, 2], [-2, 1, 2]], [8, -11, -3]))
print(solve_pp([[1e-20, 1], [1, 1]], [1, 2]))
```

```
[ 2.  3. -1.]
[1. 1.]
```

In practice call `numpy.linalg.solve`, which calls the LAPACK library's pivoted LU factorisation. Never compute A^{-1} and multiply by it: that costs about three times as much and is less accurate.

32.8.4. Iterative methods: Jacobi

For a sparse system with a million unknowns, each equation involving only a handful of neighbours (a grid in a heat-flow simulation, links in a web graph), elimination is hopeless: it fills in the zeros and costs $\sim n^3$. **Iterative methods** instead start from a guess and improve it, touching only the nonzero entries each sweep.

Jacobi's idea: solve equation i for unknown i, using the *previous* values of all the others. For

$$\begin{aligned} 4x + y &= 9 \\ x + 3y &= 5 \end{aligned} \qquad\Longrightarrow\qquad x_{k+1} = \frac{9 - y_k}{4}, \quad y_{k+1} = \frac{5 - x_k}{3}.$$

Start from $(x_0, y_0) = (0, 0)$:

$$
\begin{aligned}
x_1 &= \tfrac{9-0}{4} = 2.25, & y_1 &= \tfrac{5-0}{3} = 1.6667 \\
x_2 &= \tfrac{9-1.6667}{4} = 1.8333, & y_2 &= \tfrac{5-2.25}{3} = 0.9167 \\
x_3 &= \tfrac{9-0.9167}{4} = 2.0208, & y_3 &= \tfrac{5-1.8333}{3} = 1.0556 \\
x_4 &= 1.9861, & y_4 &= 0.9931
\end{aligned}
$$

The iterates spiral in on the exact solution $(2, 1)$; by $k = 8$ they read $(1.99990, 0.99995)$. Each sweep shrinks the error by a roughly constant factor (here about $1/\sqrt{12} \approx 0.29$): linear convergence, exactly the fixed-point iteration of the root-finding section in vector form.

When does it converge? A sufficient condition is that A is **strictly diagonally dominant**: in each row, the diagonal entry is larger in absolute value than the sum of the absolute values of the others ($4 > 1, 3 > 1$ above). Swap the equations to $x + 3y = 5,\ 4x + y = 9$ and Jacobi diverges.

Example 43. Procedure — Jacobi iteration

Input: A (ideally strictly diagonally dominant), $\mathbf{b}$, an initial guess $\mathbf{x}^{(0)}$, a tolerance.

Steps:

1. For each i, compute $x_i^{(k+1)} = \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}\right) / a_{ii}$, using only old values.
2. Stop when $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|$ (or the residual $\|A\mathbf{x}^{(k+1)} - \mathbf{b}\|$) is below tolerance, or after a maximum number of sweeps.

Output: an approximate solution $\mathbf{x}$.

Gauss–Seidel uses each new value as soon as it is computed and typically converges about twice as fast; **conjugate gradient** (for symmetric positive-definite A) and **GMRES** are the modern workhorses.

PYTHON

```python
import numpy as np

def jacobi(A, b, x0=None, tol=1e-10, max_iter=500):
    A, b = np.asarray(A, float), np.asarray(b, float)
    D = np.diag(A)                       # diagonal entries
    R = A - np.diagflat(D)               # off-diagonal part
    x = np.zeros_like(b) if x0 is None else np.asarray(x0, float)
    for k in range(1, max_iter + 1):
        x_new = (b - R @ x) / D
        if np.linalg.norm(x_new - x, np.inf) < tol:
            return x_new, k
        x = x_new
    raise RuntimeError("Jacobi did not converge")

print(jacobi([[4, 1], [1, 3]], [9, 5]))
```

```
(array([2., 1.]), 21)
```

Computer-science connection: PageRank is an iterative solver

Google's original PageRank solves a linear system with one unknown per web page (billions). Nobody eliminates it; it is solved by repeatedly multiplying a vector by a sparse link matrix (power iteration, Chapter 28), which is a Jacobi-like fixed-point iteration that converges because the damping factor 0.85 makes the system diagonally dominant in the right sense.

32.8.5. Where you'd use this

- **Electronics:** circuit simulators (SPICE and its descendants) write Kirchhoff's laws for every node of a circuit as one linear system and solve it with pivoted elimination, thousands of times per simulated microsecond.
- **Structural engineering:** the finite-element model of a bridge or aircraft wing has a linear system with millions of unknowns (the displacements of mesh points). Its matrix is sparse and symmetric, so engineers use iterative solvers in the Jacobi family rather than elimination.
- **Medicine:** a CT scanner reconstructs a cross-section of the body from X-ray measurements; each pixel is an unknown and each ray an equation. The first scanners solved this system with an iterative method (the algebraic reconstruction technique) in the same spirit as Jacobi.

US curriculum

Gaussian elimination and row reduction of augmented matrices appear in **Algebra II** or **Precalculus** (often with calculators). Pivoting strategies, conditioning and iterative solvers are university **Linear Algebra** and **Numerical Linear Algebra**.

32.9. Solving differential equations: Euler, RK4 and energy drift

32.9.1. The problem

Mechanics students learn Newton's second law: force equals mass times acceleration. For a pendulum of length L swinging through angle $\theta(t)$ this gives

$$\theta''(t) = -\frac{g}{L}\sin\theta(t).$$

For small swings, $\sin\theta \approx \theta$ and the solution is a cosine (Volume 1, Chapter 13). For large swings there is no elementary formula. Planetary orbits, epidemics, circuits and chemical reactions are the same story: an equation for the *rate of change* of a state, and no formula for the state itself. We step forward in time instead.



Differential equation, initial value problem

A **differential equation** relates an unknown function to its derivatives. An **initial value problem** (IVP) gives $\mathbf{y}'(t) = F(t, \mathbf{y})$ together with a starting value $\mathbf{y}(t_0) = \mathbf{y}_0$. Any higher-order equation becomes a first-order **system** by adding variables: for the pendulum take the state $\mathbf{y} = (\theta, \omega)$ with angular velocity $\omega = \theta'$, so $\theta' = \omega,\ \omega' = -\frac{g}{L}\sin\theta$. The **step size** h is the time increment between computed states.

32.9.2. Euler's method: follow the tangent

Intuition. $F(t, y)$ tells you the slope of the solution right now. Walk in that direction for a time h, then look again:

$$y_{n+1} = y_n + h\,F(t_n, y_n).$$

It is the tangent-line approximation of Chapter 17 applied over and over.

By hand. Take $y' = y,\ y(0) = 1$, whose exact solution is e^t, and step to $t = 1$ with $h = 0.25$: *Try it:* Try Euler with h = 0.25 .

$$\begin{aligned} y_1 &= 1 + 0.25(1) = 1.25 \\ y_2 &= 1.25 + 0.25(1.25) = 1.5625 \\ y_3 &= 1.5625 + 0.25(1.5625) = 1.953125 \\ y_4 &= 1.953125 + 0.25(1.953125) = 2.44140625 \end{aligned}$$

The true value is $e = 2.71828$; Euler is low by 0.277 because the slope keeps increasing during each step and Euler always uses the stale slope from the start. (Notice that $y_4 = 1.25^4$: Euler computes $(1 + h)^{1/h}$, the compound-interest approximation of e from Volume 1, Chapter 12 .) With $h = 0.1, 0.01, 0.001$ the errors are $0.125, 0.0135, 0.00136$: the error is proportional to h. Euler is a **first-order** method; for six correct digits you would need about a million steps.

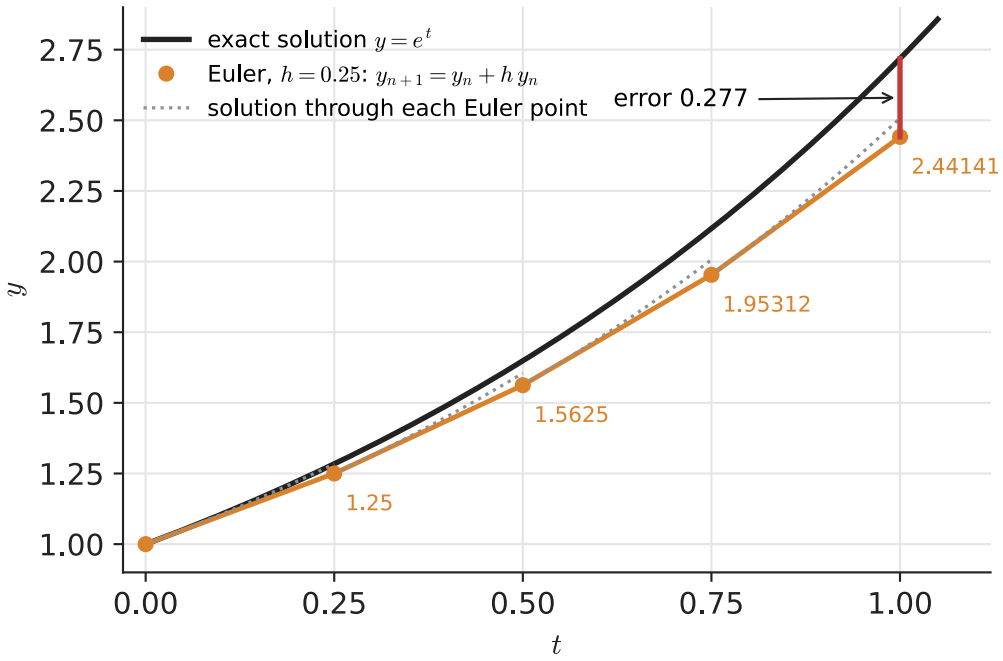


Figure 91. Euler's method for $y' = y$, $y(0) = 1$ with $h = 0.25$. Each orange segment follows the slope at the start of its step; because the true slope keeps increasing, every step lands a little low and the shortfall compounds to 0.277 at $t = 1$.

32.9.3. Runge–Kutta 4: sample the slope four times

Intuition. Instead of trusting the slope at the start of the step, probe it at the start, twice at the midpoint, and at the end, and take a weighted average, in exactly the proportions of Simpson's rule ($1:2:2:1$ is $1:4:1$ with the middle weight split):

$$\begin{aligned} k_1 &= F(t_n,\ y_n) \\ k_2 &= F\left(t_n + \tfrac{h}{2},\ y_n + \tfrac{h}{2}k_1\right) \\ k_3 &= F\left(t_n + \tfrac{h}{2},\ y_n + \tfrac{h}{2}k_2\right) \\ k_4 &= F(t_n + h,\ y_n + hk_3) \\ y_{n+1} &= y_n + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

By hand, one step of $y' = y$, $y_0 = 1$, $h = 0.25$:

$$\begin{aligned} k_1 &= 1 \\ k_2 &= 1 + 0.125(1) = 1.125 \\ k_3 &= 1 + 0.125(1.125) = 1.140625 \\ k_4 &= 1 + 0.25(1.140625) = 1.28515625 \\ y_1 &= 1 + \tfrac{0.25}{6}(1 + 2.25 + 2.28125 + 1.28515625) = 1 + \tfrac{0.25}{6}(6.81640625) = 1.2840169\ldots \end{aligned}$$

The exact value is $e^{0.25} = 1.2840254$. One RK4 step is already correct to 5 digits, where Euler's first step (1.25) was correct to 1. Four steps reach $t = 1$ with error 7.2×10^{-5}, compared to Euler's 0.277. RK4 is **fourth order**: halve h and the error falls by 16. It costs four evaluations of F per step, but that is a bargain.

Example 44. Procedure — Runge–Kutta 4 (RK4)

Input: $F(t, \mathbf{y})$, initial state $\mathbf{y}_0$ at t_0, step h, number of steps N.

Steps: for $n = 0, 1, \ldots, N-1$, compute k_1, k_2, k_3, k_4 as above (each a vector if the state is a vector), set $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ and $t_{n+1} = t_n + h$.

Output: the approximate trajectory $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N$.

32.9.4. Experiment: energy drift in a pendulum

A frictionless pendulum conserves energy: per unit mass and with $g/L = 1$, *Try it:* Run the energy-drift experiment .

$$E = \tfrac{1}{2}\omega^2 + (1 - \cos\theta)$$

(kinetic plus potential) is constant forever. Energy therefore makes a superb **diagnostic**: any drift in E is pure numerical error. Release the pendulum from rest at $\theta = 1$ radian and simulate 60 time units with $h = 0.1$, using Euler, RK4, and a third contender, the **semi-implicit (symplectic) Euler** method, which differs from Euler only in updating the velocity first and then using the *new* velocity to update the position:

PYTHON

```python
import math

def accel(theta):
    return -math.sin(theta)          # g/L = 1

def energy(theta, omega):
    return 0.5 * omega**2 + (1 - math.cos(theta))

def euler_step(th, om, h):
    return th + h * om, om + h * accel(th)

def semi_implicit_step(th, om, h):
    om = om + h * accel(th)          # update velocity first...
    return th + h * om, om           # ...then use the NEW velocity

def rk4_step(th, om, h):
    k1t, k1o = om,              accel(th)
    k2t, k2o = om + h/2 * k1o,  accel(th + h/2 * k1t)
    k3t, k3o = om + h/2 * k2o,  accel(th + h/2 * k2t)
    k4t, k4o = om + h * k3o,    accel(th + h * k3t)
    return (th + h/6 * (k1t + 2*k2t + 2*k3t + k4t),
            om + h/6 * (k1o + 2*k2o + 2*k3o + k4o))

def simulate(step, h=0.1, t_end=60.0):
    th, om = 1.0, 0.0                # released from rest at 1 radian
    E0 = energy(th, om)
    for _ in range(round(t_end / h)):
        th, om = step(th, om, h)
    return energy(th, om) / E0

for name, step in [("Euler", euler_step),
                   ("semi-implicit", semi_implicit_step),
                   ("RK4", rk4_step)]:
    print(f"{name:14s} E(60)/E(0) = {simulate(step):.6f}")
```

Try it: Run this in your browser .

```
Euler          E(60)/E(0) = 8.123523
semi-implicit  E(60)/E(0) = 1.021577
RK4            E(60)/E(0) = 0.999994
```

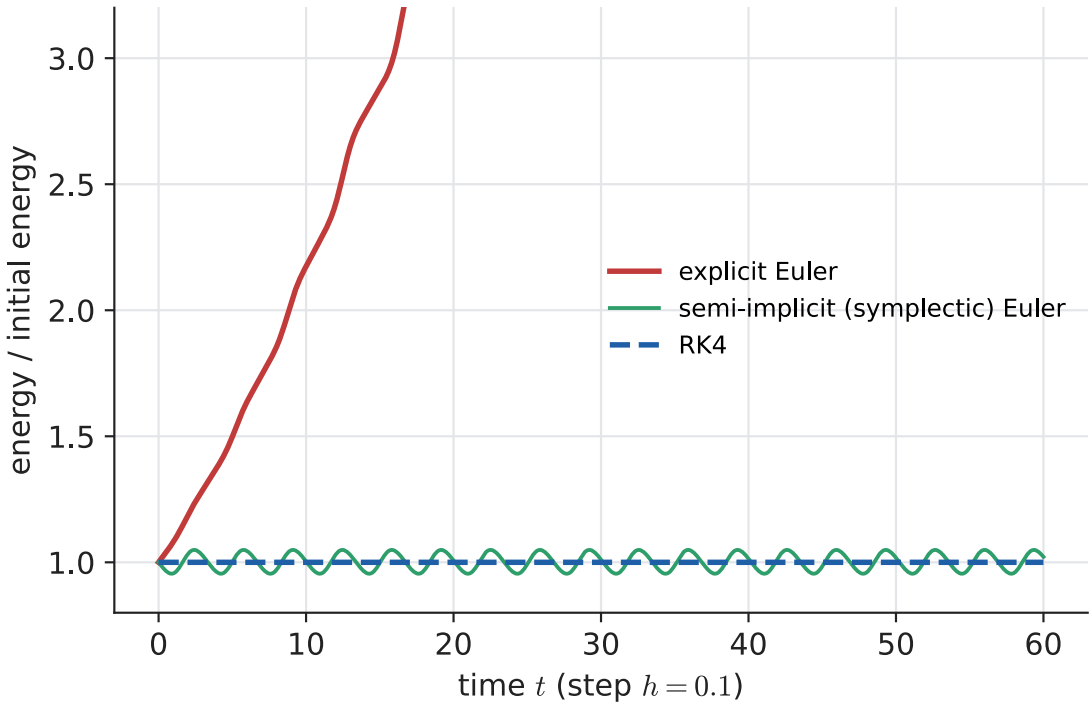


Figure 92. Energy in a simulated frictionless pendulum (step $h = 0.1$)

Three very different personalities:

- **Explicit Euler** pumps energy in at every step: after 60 time units the pendulum has eight times its starting energy and is whirling over the top. Smaller h slows the drift (with $h = 0.01$ the ratio is 1.62 at $t = 60$) but never stops it.
- **RK4** is far more accurate per step and the energy looks perfectly flat on this plot. Over very long runs it slowly *leaks* energy (ratio 0.9994 after 6000 time units): small, but systematic.
- **Semi-implicit Euler** is only first order, so its energy wobbles visibly, by up to about 5% at this step size, but the wobble stays bounded forever (still within 5% of the starting energy throughout 6000 time units). It belongs to the family of **symplectic integrators**, which exactly preserve a nearby "shadow" energy and therefore cannot drift.

The lesson: **order of accuracy is not the only virtue.** For long simulations of conservative systems (planetary orbits over millions of years, molecular dynamics, game physics), preserving the right structure beats raw accuracy per step.



Stiffness

Some systems mix very fast and very slow processes (a chemical reaction with a microsecond and an hour time scale). Explicit methods like Euler and RK4 must then use a step size dictated by the fastest process, even after it has died away, or they blow up. Such systems are **stiff** and need **implicit** methods (backward Euler, BDF), which solve an equation (often with Newton's method) at each step. If your ODE solver crawls or explodes, suspect stiffness and try `scipy.integrate.solve_ivp(…, method="BDF")`.

Computer-science connection: game physics

Most game physics engines integrate with semi-implicit Euler or **Verlet** integration (also symplectic), not RK4: they are cheap, stable for springs and orbits, and never let a swinging rope gain energy and explode. Fixed time steps are also preferred, because a step that changes with frame rate makes the simulation non-deterministic, which breaks replays and networked multiplayer.



Orbits and drift

Before symplectic methods, long-term solar-system simulations showed planets slowly spiralling into the Sun or escaping, purely from integrator error. Symplectic integrators such as Wisdom–Holman (1991) made billion-year simulations credible, and with them the discovery that the inner solar system is chaotic: Mercury's orbit cannot be predicted beyond a few tens of millions of years.



Integrator shoot-out

Animate three pendulums side by side, one per integrator, with a live energy trace underneath. Sliders control h and the release angle. Then switch to a planet orbiting a star and watch Euler's orbit spiral outward while the symplectic orbit precesses but stays closed.

Try it: Open the integrator shoot-out

32.9.5. Where you'd use this

- **Everyday: a cooling cup of tea.** Newton's law of cooling, $T' = -0.1(T - 20)$ (per minute, room at 20 °C), with tea at 90 °C. Two Euler steps of one minute: $T_1 = 90 - 0.1 \times 70 = 83$, $T_2 = 83 - 0.1 \times 63 = 76.7$. The exact answer $20 + 70e^{-0.2} \approx 77.3$ °C shows Euler's typical first-order lag; RK4 with the same step agrees to four decimal places.
- **Medicine and public health:** epidemic models (the SIR equations for susceptible, infected and recovered people) are integrated step by step to forecast hospital demand; pharmacologists integrate drug-absorption models to set dosing intervals.
- **Engineering and science:** weather and climate models, spacecraft trajectories, the charging of a capacitor in a circuit simulator, and every "physics" step in a game are numerical ODE solutions; the choice of integrator decides whether they stay stable over long runs.

US curriculum

Euler's method (by hand, a few steps) is part of the **AP Calculus BC** syllabus, along with slope fields. Runge–Kutta methods, stiffness and symplectic integration belong to university **Differential Equations** and **Numerical Analysis** courses. Pendulum and projectile modelling appears in **AP Physics C: Mechanics**.

32.10. Adding up a long list: Kahan summation

32.10.1. The problem

Add 0.1 to itself ten million times. The answer should be 1,000,000. *Try it:* Add 0.1 ten million times .

```python
s = 0.0
for _ in range(10_000_000):
    s += 0.1
print(s)
```

```
999999.9998389754
```

Not only is 0.1 inexact, but once the running sum is large each addition rounds the tiny increment to the ulp of the sum, and the rounding errors accumulate. For naive summation of n terms the error bound grows like $n\,u\sum|x_i|$: with $n = 10^7$ that is a loss of 7 of the 16 digits in the worst case.

32.10.2. Intuition: keep the change

Imagine a cash register that can only display four significant digits. The total is 1000; a customer adds a 0.4 item; the register shows 1000 again and the 0.4 vanishes. Ten such items vanish entirely: the register says 1000 when the truth is 1004. A sensible cashier would keep the lost amounts in a jar and add them back in when they became big enough to register. That jar is Kahan's **compensation** variable. *Try it:* Run the cash register .

Here is the cash register in Python, using the `decimal` module set to 4 significant digits so we can watch every step:

```python
from decimal import Decimal as D, getcontext
getcontext().prec = 4                       # a 4-digit "computer"

xs = [D(1000)] + [D("0.4")] * 10            # true sum: 1004

s = D(0)
for x in xs:
    s = s + x
print("naive:", s)

s, c = D(0), D(0)
for x in xs:
    y = x - c                               # add back what was lost last time
    t = s + y
    c = (t - s) - y                         # what was lost this time (negated)
    s = t
print("Kahan:", s)
```

```
naive: 1000
Kahan: 1004
```

Trace the first few Kahan steps after the sum reaches 1000:

term	$y = x - c$	$t = \mathrm{fl}(s + y)$	$c = (t - s) - y$	new s
0.4	$0.4 - 0 = 0.4$	$\mathrm{fl}(1000.4) = 1000$	$(1000 - 1000) - 0.4 = -0.4$	1000
0.4	$0.4 - (-0.4) = 0.8$	$\mathrm{fl}(1000.8) = 1001$	$(1001 - 1000) - 0.8 = 0.2$	1001
0.4	$0.4 - 0.2 = 0.2$	$\mathrm{fl}(1001.2) = 1001$	$(1001 - 1001) - 0.2 = -0.2$	1001
0.4	$0.4 - (-0.2) = 0.6$	$\mathrm{fl}(1001.6) = 1002$	$(1002 - 1001) - 0.6 = 0.4$	1002

c always holds the negative of the rounding error just committed (in $t - s$, the part of y that actually made it into the sum, minus y). Algebraically $(t - s) - y = 0$; in floating point it is exactly the lost low-order part. This is one of the rare places where you *want* the subtraction of nearly equal numbers, because here they are exact.

Example 45. Procedure — Kahan (compensated) summation

Input: a sequence of floating-point numbers $x_1, \ldots, x_n$.

Steps:

1. Set the sum $s \leftarrow 0$ and the compensation $c \leftarrow 0$.
2. For each x_i:
   a. $y \leftarrow x_i - c$ (correct the incoming term by the error from last time).
   b. $t \leftarrow s + y$ (low-order digits of y may be lost here).
   c. $c \leftarrow (t - s) - y$ (recover them, with a minus sign).
   d. $s \leftarrow t$.

Output: s, with error bounded by about $2u \sum |x_i|$ *independent of* n (plus a negligible $O(nu^2)$ term).

PYTHON

```python
import math

def naive_sum(xs):
    s = 0.0
    for x in xs:
        s += x
    return s

def kahan_sum(xs):
    s = 0.0          # running sum
    c = 0.0          # compensation: the low-order bits we lost last time
    for x in xs:
        y = x - c            # correct the next term by what we lost before
        t = s + y            # add; low-order digits of y may be lost here
        c = (t - s) - y      # recover what was lost (algebraically zero!)
        s = t
    return s

xs = [0.1] * 10_000_000
print(naive_sum(xs))
print(kahan_sum(xs))
print(math.fsum(xs))
```

```
999999.9998389754
1000000.0
1000000.0
```

32.10.3. Alternatives and choosing

- `math.fsum` goes further and returns the correctly rounded sum of the stored values (it tracks partial sums exactly). Use it in Python whenever accuracy matters.

- **Pairwise summation** adds in a balanced tree: error grows like $\log_2 n$ instead of n. NumPy's `np.sum` does this automatically for contiguous arrays, which is why it is usually more accurate than a Python loop.
- **Sorting** the terms by increasing magnitude before adding helps when they all have the same sign.
- For a handful of numbers of similar size, naive summation is fine. Reach for compensated summation for long sums (statistics over large datasets, long simulations accumulating time, numerical integration with many points), and for sums with mixed magnitudes.



Compilers can "optimise" Kahan away

To an optimiser that assumes real-number algebra, $c = (t - s) - y$ with $t = s + y$ simplifies to $c = 0$. C and C++ compilers with `-ffast-math` (or `-Ofast`) are allowed to do precisely that, silently turning Kahan summation back into naive summation. Never compile numerically delicate code with fast-math.



Order of addition matters

Floating-point addition is commutative ($a + b = b + a$) but **not associative**: $(10^{20} + -10^{20}) + 1 = 1$, while $10^{20} + (-10^{20} + 1) = 0$. This is why a parallel reduction on a GPU can give slightly different answers from run to run (threads finish in different orders), and why "bitwise reproducible" scientific codes go to great lengths to fix the order of summation.

32.11. Interpolation: filling in between the points

32.11.1. The problem

You have values of a function only at a few points: a table of tax brackets, a few calibration measurements, keyframes of an animation. You need values in between. **Interpolation** finds a function that passes exactly through the given points and uses it to estimate the rest.

32.11.2. Lagrange's formula: one polynomial through all the points

Through any two points there is exactly one line; through three points (with distinct x) exactly one parabola; through $n + 1$ points exactly one polynomial of degree at most n (Volume 1, Chapter 7). Lagrange's idea writes it down directly. For nodes $x_0, \ldots, x_n$, build **basis polynomials** that are 1 at their own node and 0 at all the others:

$$L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}.$$

The $\prod$ symbol is the product analogue of $\sum$: multiply the factors for every j from 0 to n except $j = i$. At $x = x_i$ every factor equals 1; at any other node x_k the factor with $j = k$ is zero. Then

$$p(x) = \sum_{i=0}^{n} y_i \, L_i(x)$$

passes through every data point, because at x_k only the k-th term survives and it equals y_k.

Worked example. Find the polynomial through $(0,1),(1,3),(2,7)$ and estimate $p(1.5)$.

$$L_0(x) = \frac{(x-1)(x-2)}{(0-1)(0-2)} = \tfrac{1}{2}(x-1)(x-2)$$
$$L_1(x) = \frac{(x-0)(x-2)}{(1-0)(1-2)} = -x(x-2)$$
$$L_2(x) = \frac{(x-0)(x-1)}{(2-0)(2-1)} = \tfrac{1}{2}x(x-1)$$

$$\begin{aligned} p(x) &= 1\cdot\tfrac{1}{2}(x^2-3x+2) + 3\cdot\big(-(x^2-2x)\big) + 7\cdot\tfrac{1}{2}(x^2-x) \\ &= \tfrac{1}{2}x^2 - \tfrac{3}{2}x + 1 - 3x^2 + 6x + \tfrac{7}{2}x^2 - \tfrac{7}{2}x && \text{(expand)} \\ &= \big(\tfrac{1}{2} - 3 + \tfrac{7}{2}\big)x^2 + \big(-\tfrac{3}{2} + 6 - \tfrac{7}{2}\big)x + 1 && \text{(collect)} \\ &= x^2 + x + 1. \end{aligned}$$

Check: $p(0)=1, p(1)=3, p(2)=7$. And $p(1.5) = 2.25 + 1.5 + 1 = 4.75$. (Directly from the basis values: $L_0(1.5) = -0.125$, $L_1(1.5) = 0.75$, $L_2(1.5) = 0.375$, so $p(1.5) = -0.125 + 2.25 + 2.625 = 4.75$.)

```python
def lagrange(xs, ys, x):
    total = 0.0
    for i, (xi, yi) in enumerate(zip(xs, ys)):
        L = 1.0
        for j, xj in enumerate(xs):
            if j != i:
                L *= (x - xj) / (xi - xj)
        total += yi * L
    return total

print(lagrange([0, 1, 2], [1, 3, 7], 1.5))
```

```
4.75
```

32.11.3. Failure mode: Runge's phenomenon

More points should mean a better fit. Carl Runge showed in 1901 that this is false. Interpolate $f(x) = 1/(1+25x^2)$ on $[-1,1]$ at equally spaced points: *Try it:* See Runge's wiggle ; Same degree, Chebyshev nodes .

degree	max error, equally spaced nodes	max error, Chebyshev nodes
5	0.43	0.56
10	1.92	0.11
15	2.11	0.083
20	59.8	0.015

With equally spaced nodes the interpolant fits beautifully in the middle and oscillates wildly near the ends, and the oscillation *grows* with the degree. The cure is either to cluster the nodes near the ends, using **Chebyshev nodes** $x_k = \cos\left(\frac{(2k+1)\pi}{2(n+1)}\right)$ (the projections onto the x-axis of equally spaced points on a semicircle), or to abandon a single high-degree polynomial altogether.

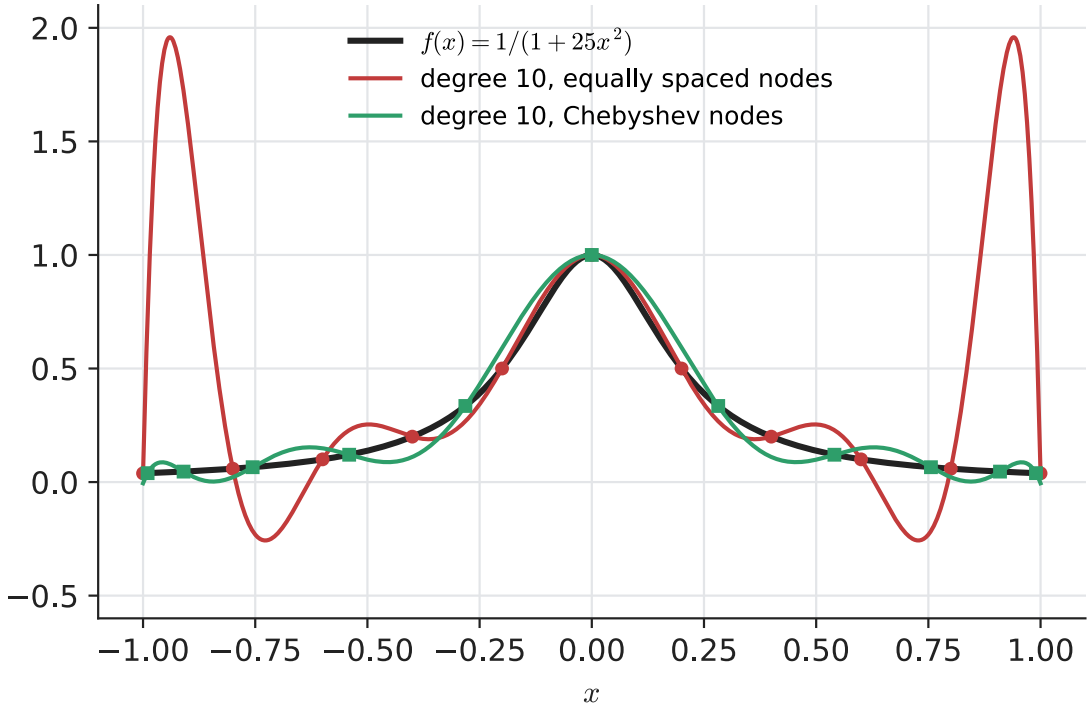


Figure 93. Runge's phenomenon: degree-10 interpolation of $1/(1+25x^2)$

32.11.4. Choosing a method

- **Piecewise linear** interpolation (join the dots): robust, never overshoots, but has corners. What `numpy.interp` does.
- **Cubic splines**: a separate cubic on each interval, joined so that the first and second derivatives match. Smooth, local, no Runge oscillation. The default for smooth curves through data.
- **Single global polynomial**: only for a few points, or with Chebyshev nodes when you get to choose where to sample.
- If the data are noisy, do not interpolate at all: **fit** (least squares, Volume 3, Chapter 31 , Volume 3, Chapter 33), because forcing a curve through every noisy point fits the noise.

Computer-science connection: curves everywhere

Fonts (TrueType uses quadratic and PostScript/OpenType uses cubic Bézier curves), vector graphics, CSS animation easing functions, camera paths in games and keyframe animation are all piecewise polynomial curves, the spline philosophy rather than the single-polynomial one. Lagrange interpolation over finite fields (Volume 3, Chapter 23) is the heart of Reed–Solomon error-correcting codes (QR codes, CDs, deep-space probes) and of Shamir's secret sharing.



Runge's wiggle

Drag the number of nodes from 2 to 25 for $1/(1+25x^2)$ and toggle between equally spaced and Chebyshev nodes, and between a global polynomial and a cubic spline. Then drag individual data points and see how far the disturbance spreads: everywhere for the global polynomial, only locally for the spline.

Try it: Open Runge's wiggle · Compare a cubic spline

32.11.5. Where you'd use this

- **Everyday:** a weather app knows forecast temperatures on a grid of points every few kilometres and every hour; the temperature "at your street at 2:30" is interpolated. If it's 14 °C at 2 pm and 18 °C at 3 pm, linear interpolation gives 16 °C at 2:30.
- **Science and engineering:** steam tables, tide tables and calibration tables for thermocouples are read by interpolation between tabulated values; a thermostat that converts sensor voltage to temperature through a lookup table interpolates every reading.
- **Computing:** animation tweening, texture sampling in graphics (bilinear interpolation between texels), and audio resampling (changing 44.1 kHz to 48 kHz) are interpolation running millions of times a second.

US curriculum

Linear interpolation in tables appears in **Algebra I**; fitting a polynomial through given points appears in **Algebra II** and **Precalculus** (often as solving a system for the coefficients). Lagrange form, splines and Runge's phenomenon are university **Numerical Analysis**.

32.12. When to trust a number

Every section of this chapter has shown a way in which a confident-looking 16-digit number can be wrong. Pulling the lessons together gives a practical discipline.

32.12.1. Sources of error, in one place

Source	Example	What controls it
Modelling error	$\sin\theta \approx \theta$; frictionless pendulum	the science, not the numerics
Data error	measured inputs good to 3 digits	conditioning: expect about $\kappa\times$ data error
Truncation (discretisation) error	finite h, finitely many terms or iterations	the method's order; refine h
Rounding error	53-bit arithmetic	stability of the algorithm; avoid cancellation
Bugs	wrong sign, off-by-one in a loop	tests against known answers

The order matters. There is no point computing to 15 digits a model that is right to 2, or polishing rounding error when the step size is the dominant error.

Example 46. Procedure — Should I believe this number?

Input: a computed result.

Steps:

1. **Sanity:** Are the units, sign and order of magnitude right? Do conserved quantities stay conserved? Does a probability lie in $[0, 1]$?
2. **Known cases:** Does the same code reproduce an answer you know exactly (a test problem like $x^2 - 2$, $\int_0^1 e^x\,dx$, a linear pendulum)?
3. **Convergence:** Halve the step size (or double the points, or tighten the tolerance). Does the answer change in the digits you care about? Does the change shrink at the rate the method's order predicts?
4. **Residual:** For an equation, plug the answer back in. For $A\mathbf{x} = \mathbf{b}$, is $\|A\mathbf{x} - \mathbf{b}\|$ small? (A small residual with a large condition number still permits a large error.)
5. **Conditioning:** Perturb the inputs by their uncertainty. How much does the output move?
6. **Independent method:** Solve it another way (a different algorithm, a different library, higher precision with `decimal` or `mpmath`). Agreement between genuinely different methods is strong evidence.

Output: a result together with an honest estimate of how many of its digits mean something.



Precision is not accuracy

A double always prints with up to 17 significant digits. That says nothing about how many are correct. The forward recurrence for I_{20} printed `-30.19239488558378` with total confidence. Report results with the number of digits you have justified, not the number the machine printed.

Computer-science connection: interval arithmetic and verified computing

Interval arithmetic carries a lower and upper bound through every operation, rounding the lower bound down and the upper bound up (IEEE 754 provides those directed rounding modes for exactly this purpose). The final interval is *guaranteed* to contain the true answer. It was used in the computer-assisted proof of the Kepler conjecture on sphere packing and in proving that the Lorenz attractor exists. Its weakness is pessimism: intervals can grow far wider than the real error.

32.13. Interactive tools

Browser tools on the companion website that go with this chapter. Each example link in the chapter opens one of them with that example already set up.

- Floating-point inspector
- Where floating point bites
- Linear systems, visually
- Root finding, step by step
- ODE solver and phase portraits
- Language sandbox

32.14. Exercises

1. **Recognition.** For each, name the kind of error that dominates (rounding, cancellation, truncation, overflow, underflow, instability, ill-conditioning): (a) `1e308 * 10`; (b) multiplying 500 probabilities of 0.1; (c) forward difference with $h = 10^{-14}$; (d) Euler's method with $h = 0.5$; (e) solving two nearly parallel lines from data measured to 3 digits.
2. Convert 0.75 and 0.3 to binary. Which is stored exactly as a double, and why?
3. What is $\mathrm{ulp}(2^{30})$ for a double? Can a double represent $2^{30} + 0.25$ exactly?
4. Evaluate $\sqrt{x^2+1} - x$ for $x = 10^8$ naively in Python, then derive and test a cancellation-free rewrite.
5. Compute the condition number $\kappa(x) = |x f'(x)/f(x)|$ of $f(x) = \ln x$. Where is taking a logarithm ill-conditioned?

6. Perform three bisection steps and three Newton steps (from $x_0 = 1$) on $f(x) = x^3 - 2$. How many correct digits does each have?
7. The fixed-point iteration $x_{n+1} = \cos x_n$ solves $x = \cos x$. Using $g'(x) = -\sin x$ and the root $r \approx 0.739$, predict the convergence ratio, then check by experiment.
8. For $f(x) = e^x$ at $x = 0$, the forward difference with $h = 0.1$ gives what value and what error? What does the error theory predict ($\approx \frac{h}{2} f''(x)$)?
9. Apply the trapezoid rule and Simpson's rule with $n = 2$ to $\int_0^2 x^3\,dx$. Explain why Simpson's answer is exact.
10. Solve by Gaussian elimination with partial pivoting, showing each swap and multiplier: $x + 2y = 5,\ 3x + 4y = 11$.
11. **Problem solving.** Use one step of Euler and one step of RK4 with $h = 0.1$ on $y' = -2y,\ y(0) = 1$. Compare with $e^{-0.2}$. Then explain what happens to Euler if $h = 1.5$ and why (look at the multiplier per step).
12. **Problem solving.** Write a function that sums a list both naively and with Kahan's algorithm, and find a list of ten numbers for which the naive float sum is off by at least 1. (Hint: mix 10^{16} and 1.)

▼ **Answers**

1. (a) overflow, result `inf`; (b) underflow: 10^{-500} is below 4.9×10^{-324}, result 0.0 (use logs); (c) cancellation (rounding error amplified by $1/h$); (d) truncation; (e) ill-conditioning.
2. $0.75 = 0.11_2$ (exact, since $0.75 = 3/4$ has a power-of-two denominator). 0.3: $0.3 \to 0.6(0) \to 1.2(1) \to 0.4(0) \to 0.8(0) \to 1.6(1) \to 1.2(1)\ldots$, so $0.3 = 0.0\overline{1001}_2$, which never terminates (denominator 10 contains 5); it is rounded.
3. $2^{30} = 1.0 \times 2^{30}$, so $\mathrm{ulp} = 2^{30} \times 2^{-52} = 2^{-22} \approx 2.4 \times 10^{-7}$. Since $0.25 = 2^{-2}$ is a multiple of 2^{-22}, $2^{30} + 0.25$ is exact.
4. Naively, `math.sqrt(1e16+1) - 1e8` gives `0.0` ($10^{16} + 1$ rounds to 10^{16}). Multiply by the conjugate: $\sqrt{x^2+1} - x = \dfrac{(x^2+1) - x^2}{\sqrt{x^2+1} + x} = \dfrac{1}{\sqrt{x^2+1} + x}$, which gives `5e-09`, correct to full precision (true value $4.99999999999999\ldots \times 10^{-9}$).
5. $f'(x) = 1/x$, so $\kappa = |x \cdot \frac{1}{x} / \ln x| = 1/|\ln x|$. It blows up as $x \to 1$ (where $\ln x \to 0$): computing $\ln x$ for x near 1 is ill-conditioned, which is why `log1p` exists.
6. Bisection on $[1, 2]$: $m = 1.5$ ($f = 1.375 > 0$, $b = 1.5$); $m = 1.25$ ($f = -0.047 < 0$, $a = 1.25$); $m = 1.375$ ($f = 0.600 > 0$). Bracket $[1.25, 1.375]$, estimate 1.3125, about 1 correct digit ($\sqrt[3]{2} = 1.259921$). Newton: $x_{n+1} = x_n - (x_n^3 - 2)/(3x_n^2)$: $x_1 = 1 + 1/3 = 1.3333$, $x_2 = 1.3333 - (2.3704 - 2)/5.3333 = 1.2639$, $x_3 = 1.2599335$, correct to 4–5 digits (error 1.2×10^{-5}).
7. $|g'(r)| = \sin(0.739) \approx 0.674$. Each step multiplies the error by about 0.67, so about 0.17 digits per step; about 55 iterations from $x_0 = 1$ to reach 10^{-10}. Experiment confirms successive error ratios near -0.674 (the sign alternates, so the iterates zig-zag around the root).

8. $(e^{0.1}-1)/0.1 = 1.0517092$, error 0.0517. Prediction $\frac{0.1}{2}e^0 = 0.05$; the remaining 0.0017 is the next Taylor term $\frac{h^2}{6}$.

9. $h = 1$, values $0, 1, 8$. Trapezoid: $1 \cdot (0+1+4) = 5$. Simpson: $\frac{1}{3}(0+4+8) = 4$. Exact: $\int_0^2 x^3 dx = 4$. Simpson is exact for cubics because its error term involves $f^{(4)}$, which is zero.

10. Column 1: largest is 3, swap: $3x + 4y = 11,\ x + 2y = 5$. Multiplier $m = 1/3$: row 2 becomes $(2 - \frac{4}{3})y = 5 - \frac{11}{3}$, i.e. $\frac{2}{3}y = \frac{4}{3}$, so $y = 2$. Back substitute: $3x + 8 = 11$, $x = 1$.

11. Euler: $y_1 = 1 + 0.1(-2) = 0.8$. RK4: $k_1 = -2$, $k_2 = -2(1-0.1) = -1.8$, $k_3 = -2(1-0.09) = -1.82$, $k_4 = -2(1-0.182) = -1.636$; $y_1 = 1 + \frac{0.1}{6}(-2 - 3.6 - 3.64 - 1.636) = 1 - \frac{1.0876}{6} = 0.818733$. Exact $e^{-0.2} = 0.818731$. Euler error 0.019, RK4 error 2.6×10^{-6}. With $h = 1.5$ Euler multiplies by $1 + h(-2) = -2$ each step: $1, -2, 4, -8, \ldots$, an oscillating explosion although the true solution decays. Euler is stable for this equation only when $|1 - 2h| < 1$, i.e. $h < 1$. This is stiffness in miniature.

12. Example: `[1e16, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, -1e16]`. True sum 8. Naive: each `1e16 + 1.0` rounds back to `1e16` (ulp at 10^{16} is 2, and ties round to even), so the result is `0.0`. Kahan returns `8.0`.

32.15. Mastery check

1. Explain in two sentences why `0.1 + 0.2 != 0.3` in IEEE 754 doubles.
2. State machine epsilon for doubles and what the model $\mathrm{fl}(x \circ y) = (x \circ y)(1+\delta)$ says.
3. The expression $(1 - \cos x)/x^2$ should approach $\frac{1}{2}$ as $x \to 0$. What does it return for $x = 10^{-8}$ in double precision, why, and how do you fix it?
4. Distinguish an ill-conditioned problem from an unstable algorithm, with one example of each.
5. Rank bisection, secant, Newton and fixed-point iteration by order of convergence, and give one situation where you would choose bisection anyway.
6. Why does the error of a forward-difference derivative first decrease and then increase as $h \to 0$? Roughly what h is optimal in double precision?
7. Simpson's rule with n strips has error 0.0008. Estimate the error with $2n$ strips.
8. In a long orbital simulation, which would you prefer, RK4 or a symplectic method, and why?

Decision rule: All correct without hesitation → move on to Volume 3, Chapter 33 . Stuck on 1–3 → reread Section 32.2 and Section 32.3. Stuck on 4–5 → reread Section 32.4 and Section 32.5. Stuck on 6–8 → reread Section 32.6, Section 32.7 and Section 32.9, then work the exercises in Section 32.14.

▼ **Mastery check answers**

1. 0.1, 0.2 and 0.3 have infinite binary expansions, so each is stored as the nearest double, with a tiny error. The rounding errors of 0.1 and 0.2 are both upward and their sum rounds to the double just above the stored 0.3.
2. $\varepsilon_{\text{mach}} = 2^{-52} \approx 2.22 \times 10^{-16}$. Each basic operation returns the exact result rounded to nearest, so it is wrong by a relative amount at most $u = 2^{-53}$.
3. $\cos(10^{-8})$ rounds to exactly 1, so the numerator is 0 and the result is 0.0 instead of 0.5: catastrophic cancellation. Rewrite as $2\sin^2(x/2)/x^2$, which gives 0.5.
4. Ill-conditioned: the answer is sensitive to the data whatever algorithm you use (nearly parallel lines; $\kappa \approx 4 \times 10^4$). Unstable: the algorithm amplifies rounding errors on a problem that is fine (the forward recurrence $I_n = 1 - nI_{n-1}$; naive elimination without pivoting on the 10^{-20} system).
5. Newton (2) > secant (about 1.618) > bisection and fixed-point (both 1; bisection's ratio is $\frac{1}{2}$, fixed-point's is $|g'(r)|$). Choose bisection when you have a sign-change bracket and need guaranteed convergence with no derivative, e.g. a one-off root of a black-box function, or as the safeguard inside Brent's method.
6. Truncation error (proportional to h) falls as h shrinks, but cancellation in $f(x+h) - f(x)$ produces a rounding error of about $u|f|/h$ that grows. The optimum is near $\sqrt{u} \approx 10^{-8}$ (about 10^{-5} to 10^{-6} for central differences).
7. Simpson is fourth order: doubling n halves h and divides the error by $2^4 = 16$, giving about 5×10^{-5}.
8. A symplectic method (semi-implicit Euler, Verlet, Wisdom–Holman). RK4 is more accurate per step but its energy drifts systematically over long times, so orbits slowly decay or spread; symplectic methods keep energy errors bounded forever.

Appendix A: Cheat Sheet

This is the "I remember learning this — remind me what it means" appendix. Every row is a reminder, not a lesson: if a formula looks unfamiliar, follow the chapter link and read the explanation in context. Definitions of terms live in the Glossary (Appendix B); symbols and how to read them aloud are in Appendix C.

Conventions used throughout: letters $a, b, c, \ldots$ are real numbers unless stated otherwise; denominators are assumed nonzero; logarithms and square roots are applied only where they are defined; $\ln$ is the natural logarithm (base e); angles in calculus formulas are in radians; C is an arbitrary constant of integration.

A.1. Arithmetic and fractions

Rule	Formula or statement	Ch.
Order of operations	Brackets, then powers, then $\times$/$\div$ left to right, then $+$/$-$ left to right. $2 + 3 \times 4^2 = 2 + 48 = 50$	Volume 1, Chapter 1
Signs	$(-a)(-b) = ab$, $(-a)b = -ab$, $a - (-b) = a + b$	Volume 1, Chapter 1
Division with remainder	$a = qb + r$ with $0 \le r < b$ (quotient q, remainder r)	Volume 1, Chapter 1
Absolute value	$\lvert x \rvert = x$ if $x \ge 0$, $-x$ if $x < 0$; distance between a and b is $\lvert a - b \rvert$	Volume 1, Chapter 1
Place value in base b	$(d_k \ldots d_1 d_0)_b = \sum_{i=0}^{k} d_i \, b^i$, e.g. $1011_2 = 8 + 0 + 2 + 1 = 11$	Volume 1, Chapter 1
gcd and lcm	$\gcd(a, b) \cdot \operatorname{lcm}(a, b) = ab$ for positive integers	Volume 1, Chapter 1
Divisibility tests	By 3 or 9: digit sum divisible by 3 or 9. By 11: alternating digit sum divisible by 11. By 4: last two digits divisible by 4.	Volume 1, Chapter 1
Equivalent fractions	$\frac{a}{b} = \frac{ka}{kb}$ for $k \neq 0$	Volume 1, Chapter 2

Rule	Formula or statement	Ch.
Add / subtract fractions	$\frac{a}{b} \pm \frac{c}{d} = \frac{ad \pm bc}{bd}$ (or use the least common denominator)	Volume 1, Chapter 2
Multiply fractions	$\frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}$	Volume 1, Chapter 2
Divide fractions	$\frac{a}{b} \div \frac{c}{d} = \frac{a}{b} \cdot \frac{d}{c} = \frac{ad}{bc}$	Volume 1, Chapter 2
Percent of a quantity	$p\%$ of x is $\frac{p}{100}\,x$	Volume 1, Chapter 2
Percentage change	$\frac{\text{new} - \text{old}}{\text{old}} \times 100\%$	Volume 1, Chapter 2
Successive changes	Multiply factors: up 10% then down 10% is $1.1 \times 0.9 = 0.99$, a 1% fall overall	Volume 1, Chapter 2
Ratio split	Sharing T in the ratio $a : b$ gives $\frac{a}{a+b}T$ and $\frac{b}{a+b}T$	Volume 1, Chapter 2
Repeating decimal to fraction	$0.\overline{d_1 \ldots d_k} = \frac{d_1 \ldots d_k}{10^k - 1}$, e.g. $0.\overline{27} = \frac{27}{99} = \frac{3}{11}$	Volume 1, Chapter 2

A.2. Exponents, roots, and logarithms

Rule	Formula	Ch.
Product of powers	$a^m a^n = a^{m+n}$	Volume 1, Chapter 3
Quotient of powers	$\frac{a^m}{a^n} = a^{m-n}$	Volume 1, Chapter 3
Power of a power	$(a^m)^n = a^{mn}$	Volume 1, Chapter 3
Power of a product / quotient	$(ab)^n = a^n b^n$, $\left(\frac{a}{b}\right)^n = \frac{a^n}{b^n}$	Volume 1, Chapter 3
Zero and negative exponents	$a^0 = 1$ $(a \neq 0)$, $a^{-n} = \frac{1}{a^n}$	Volume 1, Chapter 3
Fractional exponents	$a^{1/n} = \sqrt[n]{a}$, $a^{m/n} = \left(\sqrt[n]{a}\right)^m = \sqrt[n]{a^m}$ $(a > 0)$	Volume 1, Chapter 3
Root rules	$\sqrt{ab} = \sqrt{a}\sqrt{b}$, $\sqrt{a/b} = \sqrt{a}/\sqrt{b}$ for $a, b \geq 0$; but $\sqrt{a+b} \neq \sqrt{a} + \sqrt{b}$	Volume 1, Chapter 3
Square root of a square	$\sqrt{x^2} = \lvert x \rvert$	Volume 1, Chapter 3

Rule	Formula	Ch.
Rationalising	$\dfrac{1}{\sqrt{a}} = \dfrac{\sqrt{a}}{a}$, $\dfrac{1}{\sqrt{a}+\sqrt{b}} = \dfrac{\sqrt{a}-\sqrt{b}}{a-b}$	Volume 1, Chapter 3
Scientific notation	$x = m \times 10^n$ with $1 \le \lvert m \rvert < 10$; multiply by multiplying mantissas and adding exponents	Volume 1, Chapter 3
Definition of logarithm	$\log_b x = y \iff b^y = x$ $(b > 0, b \neq 1, x > 0)$	Volume 1, Chapter 12
Product / quotient laws	$\log_b(xy) = \log_b x + \log_b y$, $\log_b(x/y) = \log_b x - \log_b y$	Volume 1, Chapter 12
Power law	$\log_b(x^k) = k \log_b x$	Volume 1, Chapter 12
Special values	$\log_b 1 = 0$, $\log_b b = 1$, $b^{\log_b x} = x$, $\log_b(b^x) = x$	Volume 1, Chapter 12
Change of base	$\log_b x = \dfrac{\ln x}{\ln b} = \dfrac{\log_{10} x}{\log_{10} b}$; e.g. $\log_2 x \approx 3.32 \log_{10} x$	Volume 1, Chapter 12
Exponential via e	$a^x = e^{x \ln a}$	Volume 1, Chapter 12

Rule	Formula	Ch.
Not a law	$\log(x+y) \neq \log x + \log y$; $(\log x)^2 \neq \log(x^2)$	Volume 1, Chapter 12
Exponential growth / decay	$N(t) = N_0 e^{kt}$ (growth $k > 0$, decay $k < 0$); half-life form $N_0\left(\frac{1}{2}\right)^{t/T}$	Volume 1, Chapter 12
Doubling time	$T_2 = \frac{\ln 2}{k}$; for $r\%$ per period, roughly $70/r$ periods ("rule of 70")	Volume 1, Chapter 12
Compound interest	$A = P\left(1 + \frac{r}{n}\right)^{nt}$; continuously compounded $A = Pe^{rt}$	Volume 1, Chapter 12
Defining e	$e = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n \approx 2.71828$	Volume 1, Chapter 12

A.3. Algebraic identities and polynomials

Identity	Formula	Ch.
Distributive law	$a(b+c) = ab + ac$	Volume 1, Chapter 4
Expanding two brackets	$(a+b)(c+d) = ac + ad + bc + bd$	Volume 1, Chapter 4
Square of a sum / difference	$(a \pm b)^2 = a^2 \pm 2ab + b^2$	Volume 1, Chapter 4
Difference of two squares	$a^2 - b^2 = (a+b)(a-b)$	Volume 1, Chapter 4
Cube of a sum / difference	$(a \pm b)^3 = a^3 \pm 3a^2b + 3ab^2 \pm b^3$	Volume 1, Chapter 7
Sum / difference of cubes	$a^3 + b^3 = (a+b)(a^2 - ab + b^2)$, $a^3 - b^3 = (a-b)(a^2 + ab + b^2)$	Volume 1, Chapter 7
Difference of nth powers	$a^n - b^n = (a-b)(a^{n-1} + a^{n-2}b + \cdots + ab^{n-2} + b^{n-1})$	Volume 1, Chapter 7
Binomial theorem	$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k}b^k$	Volume 3, Chapter 24

Identity	Formula	Ch.
Remainder theorem	Dividing $p(x)$ by $(x - r)$ leaves remainder $p(r)$	Volume 1, Chapter 7
Factor theorem	$(x - r)$ is a factor of $p(x) \iff p(r) = 0$	Volume 1, Chapter 7
Rational root theorem	If p/q (lowest terms) is a root of $a_n x^n + \cdots + a_0$ with integer coefficients, then p divides a_0 and q divides a_n	Volume 1, Chapter 7
Number of roots	A degree-n polynomial has at most n real roots, and exactly n complex roots counted with multiplicity	Volume 1, Chapter 7
End behaviour	Determined by the leading term $a_n x^n$: its sign and whether n is even or odd	Volume 1, Chapter 7
Simplifying rational expressions	Cancel common *factors*, never common *terms*: $\frac{x(x+1)}{x} = x + 1$ for $x \neq 0$, but $\frac{x+1}{x} \neq 1$	Volume 1, Chapter 4

A.4. Quadratics

Fact	Formula	Ch.
Quadratic formula	$ax^2+bx+c=0 \implies x=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$ ($a\neq 0$)	Volume 1, Chapter 8
Discriminant	$\Delta=b^2-4ac$: $\Delta>0$ two real roots; $\Delta=0$ one repeated real root; $\Delta<0$ two complex-conjugate roots	Volume 1, Chapter 8
Completing the square	$x^2+bx=\left(x+\frac{b}{2}\right)^2-\frac{b^2}{4}$	Volume 1, Chapter 8
Vertex form	$ax^2+bx+c=a(x-h)^2+k$ with $h=-\dfrac{b}{2a}$, $k=c-\dfrac{b^2}{4a}$; vertex (h,k), axis of symmetry $x=h$	Volume 1, Chapter 8
Factored form	$a(x-r_1)(x-r_2)$ where r_1, r_2 are the roots	Volume 1, Chapter 8
Sum and product of roots (Vieta)	$r_1+r_2=-\dfrac{b}{a}$, $r_1 r_2=\dfrac{c}{a}$	Volume 1, Chapter 8
Numerically stable roots	$q=-\frac{1}{2}\left(b+\operatorname{sign}(b)\sqrt{\Delta}\right)$, then $x_1=q/a$, $x_2=c/q$ (avoids cancellation when $b^2\gg 4ac$)	Volume 1, Chapter 8

A.5. Linear equations, lines, and coordinates

Fact	Formula	Ch.
Inequalities	Multiplying or dividing both sides by a negative number reverses the inequality: $-2x < 6 \iff x > -3$	Volume 1, Chapter 5
Absolute-value inequality	$\lvert x - a \rvert < r \iff a - r < x < a + r$	Volume 1, Chapter 5
Slope	$m = \dfrac{y_2 - y_1}{x_2 - x_1} = \dfrac{\text{rise}}{\text{run}}$	Volume 1, Chapter 5
Slope-intercept form	$y = mx + b$ (slope m, y-intercept b)	Volume 1, Chapter 5
Point-slope form	$y - y_1 = m(x - x_1)$	Volume 1, Chapter 5
Standard form	$Ax + By = C$; slope $-A/B$	Volume 1, Chapter 5
Parallel / perpendicular	Parallel: $m_1 = m_2$. Perpendicular: $m_1 m_2 = -1$	Volume 1, Chapter 5
Two equations, two unknowns (Cramer)	$ax + by = e,\ cx + dy = f \implies x = \dfrac{ed - bf}{ad - bc},\ y = \dfrac{af - ec}{ad - bc}$ (if $ad - bc \neq 0$)	Volume 1, Chapter 6

Fact	Formula	Ch.
Number of solutions of a linear system	Exactly one, none (inconsistent), or infinitely many; never exactly two	Volume 1, Chapter 6
Distance	$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$	Volume 1, Chapter 10
Midpoint	$\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$	Volume 1, Chapter 10
Circle	Centre (h, k), radius r: $(x - h)^2 + (y - k)^2 = r^2$	Volume 1, Chapter 10
Point-to-line distance	From (x_0, y_0) to $Ax + By + C = 0$: $\frac{\lvert Ax_0 + By_0 + C\rvert}{\sqrt{A^2 + B^2}}$	Volume 1, Chapter 10
Transformations of graphs	$y = f(x - h) + k$ shifts right h, up k; $y = af(x)$ stretches vertically by a; $y = f(bx)$ compresses horizontally by b; $y = -f(x)$ reflects in the x-axis; $y = f(-x)$ in the y-axis	Volume 1, Chapter 11
Composition and inverse	$(f \circ g)(x) = f(g(x))$; $f^{-1}(f(x)) = x$; graph of f^{-1} is the reflection of the graph of f in $y = x$	Volume 1, Chapter 11

A.6. Geometry

Shape or fact	Formula	Ch.
Pythagoras	Right triangle with legs a, b, hypotenuse c: $a^2 + b^2 = c^2$. Triples: 3-4-5, 5-12-13, 8-15-17	Volume 1, Chapter 9
Angle sums	Triangle $180°$; n-sided polygon $(n-2) \times 180°$; exterior angles of a convex polygon total $360°$	Volume 1, Chapter 9
Special right triangles	45-45-90: sides $1 : 1 : \sqrt{2}$. 30-60-90: sides $1 : \sqrt{3} : 2$	Volume 1, Chapter 9
Rectangle / parallelogram	Area lw; parallelogram bh (h perpendicular height)	Volume 1, Chapter 9
Triangle	Area $\frac{1}{2}bh = \frac{1}{2}ab\sin C$; Heron: $\sqrt{s(s-a)(s-b)(s-c)}$ with $s = \frac{a+b+c}{2}$	Volume 1, Chapter 9
Trapezoid (UK: trapezium)	Area $\frac{1}{2}(a+b)h$, a, b the parallel sides	Volume 1, Chapter 9
Circle	Circumference $2\pi r = \pi d$; area πr^2	Volume 1, Chapter 9
Arc and sector (radians)	Arc length $s = r\theta$; sector area $\frac{1}{2}r^2\theta$	Volume 1, Chapter 9

Shape or fact	Formula	Ch.
Degrees and radians	$\pi \text{ rad} = 180°$; $1 \text{ rad} \approx 57.3°$	Volume 1, Chapter 13
Prism / cylinder	Volume Bh (B = base area); cylinder $\pi r^2 h$, surface area $2\pi r^2 + 2\pi r h$	Volume 1, Chapter 9
Pyramid / cone	Volume $\frac{1}{3}Bh$; cone $\frac{1}{3}\pi r^2 h$	Volume 1, Chapter 9
Sphere	Volume $\frac{4}{3}\pi r^3$; surface area $4\pi r^2$	Volume 1, Chapter 9
Similar figures	Scale lengths by k: areas scale by k^2, volumes by k^3	Volume 1, Chapter 9
Circle theorems	Angle in a semicircle is $90°$; inscribed angle is half the central angle on the same arc; tangent is perpendicular to the radius	Volume 1, Chapter 9

A.7. Trigonometry

Fact	Formula	Ch.
SOH-CAH-TOA	$\sin\theta = \frac{\text{opp}}{\text{hyp}}, \cos\theta = \frac{\text{adj}}{\text{hyp}}, \tan\theta = \frac{\text{opp}}{\text{adj}} = \frac{\sin\theta}{\cos\theta}$	Volume 1, Chapter 13
Reciprocal functions	$\sec\theta = \frac{1}{\cos\theta}, \csc\theta = \frac{1}{\sin\theta}, \cot\theta = \frac{1}{\tan\theta}$	Volume 1, Chapter 13
Unit circle	The point at angle θ is $(\cos\theta, \sin\theta)$	Volume 1, Chapter 13
Signs by quadrant ("All Students Take Calculus")	Q1 all positive; Q2 only $\sin$; Q3 only $\tan$; Q4 only $\cos$	Volume 1, Chapter 13
Pythagorean identities	$\sin^2\theta + \cos^2\theta = 1, 1 + \tan^2\theta = \sec^2\theta,$ $1 + \cot^2\theta = \csc^2\theta$	Volume 1, Chapter 13
Symmetry	$\sin(-\theta) = -\sin\theta, \cos(-\theta) = \cos\theta,$ $\tan(-\theta) = -\tan\theta$	Volume 1, Chapter 13
Co-function	$\sin\left(\frac{\pi}{2} - \theta\right) = \cos\theta, \cos\left(\frac{\pi}{2} - \theta\right) = \sin\theta$	Volume 1, Chapter 13
Periodicity	$\sin$ and $\cos$ have period 2π; $\tan$ has period π; $A\sin(bx + c) + d$ has amplitude $\lvert A\rvert$, period $2\pi/\lvert b\rvert$	Volume 1, Chapter 13

Fact	Formula	Ch.
Angle addition	$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$; $\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$	Volume 1, Chapter 13
Tangent addition	$\tan(\alpha \pm \beta) = \dfrac{\tan\alpha \pm \tan\beta}{1 \mp \tan\alpha\tan\beta}$	Volume 1, Chapter 13
Double angle	$\sin 2\theta = 2\sin\theta\cos\theta$; $\cos 2\theta = \cos^2\theta - \sin^2\theta = 2\cos^2\theta - 1 = 1 - 2\sin^2\theta$; $\tan 2\theta = \dfrac{2\tan\theta}{1 - \tan^2\theta}$	Volume 1, Chapter 13
Power reduction	$\sin^2\theta = \frac{1}{2}(1 - \cos 2\theta)$, $\cos^2\theta = \frac{1}{2}(1 + \cos 2\theta)$	Volume 1, Chapter 13
Law of sines	$\dfrac{a}{\sin A} = \dfrac{b}{\sin B} = \dfrac{c}{\sin C}$ ($= 2R$, R the circumradius)	Volume 1, Chapter 13
Law of cosines	$c^2 = a^2 + b^2 - 2ab\cos C$	Volume 1, Chapter 13
Inverse function ranges	$\arcsin : [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\arccos : [0, \pi]$, $\arctan : (-\frac{\pi}{2}, \frac{\pi}{2})$	Volume 1, Chapter 13
Polar coordinates	$x = r\cos\theta$, $y = r\sin\theta$; $r = \sqrt{x^2 + y^2}$, $\theta = \operatorname{atan2}(y, x)$	Volume 1, Chapter 13

Fact	Formula	Ch.
Rotation by θ	$(x, y) \mapsto (x\cos\theta - y\sin\theta,\ x\sin\theta + y\cos\theta)$	Volume 1, Chapter 13

Table 1. Exact values on the unit circle

Degrees	Radians	sin	cos	tan
$0°$	0	0	1	0
$30°$	$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{3}} = \frac{\sqrt{3}}{3}$
$45°$	$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1
$60°$	$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$
$90°$	$\frac{\pi}{2}$	1	0	undefined
$120°$	$\frac{2\pi}{3}$	$\frac{\sqrt{3}}{2}$	$-\frac{1}{2}$	$-\sqrt{3}$
$135°$	$\frac{3\pi}{4}$	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$	-1
$150°$	$\frac{5\pi}{6}$	$\frac{1}{2}$	$-\frac{\sqrt{3}}{2}$	$-\frac{\sqrt{3}}{3}$
$180°$	π	0	-1	0
$270°$	$\frac{3\pi}{2}$	-1	0	undefined
$360°$	2π	0	1	0

Memory aid for the first quadrant: $\sin$ at $0°, 30°, 45°, 60°, 90°$ is $\frac{\sqrt{0}}{2}, \frac{\sqrt{1}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{3}}{2}, \frac{\sqrt{4}}{2}$; $\cos$ is the same list backwards. See Volume 1, Chapter 13 .

A.8. Sequences and series

Fact	Formula	Ch.
Arithmetic sequence	$a_n = a_1 + (n-1)d$; sum $S_n = \dfrac{n(a_1 + a_n)}{2} = \dfrac{n}{2}(2a_1 + (n-1)d)$	Volume 1, Chapter 14
Geometric sequence	$a_n = a_1 r^{n-1}$; sum $S_n = a_1 \dfrac{1 - r^n}{1 - r}$ $(r \neq 1)$	Volume 1, Chapter 14
Infinite geometric series	$\sum_{n=0}^{\infty} ar^n = \dfrac{a}{1-r}$ if $\lvert r\rvert < 1$; diverges otherwise (for $a \neq 0$)	Volume 1, Chapter 14
Sum of first n integers	$\sum_{k=1}^{n} k = \dfrac{n(n+1)}{2}$	Volume 1, Chapter 14
Sum of squares / cubes	$\sum_{k=1}^{n} k^2 = \dfrac{n(n+1)(2n+1)}{6}$, $\sum_{k=1}^{n} k^3 = \left(\dfrac{n(n+1)}{2}\right)^2$	Volume 1, Chapter 14
Powers of two	$\sum_{k=0}^{n-1} 2^k = 2^n - 1$	Volume 1, Chapter 14
Harmonic numbers	$H_n = \sum_{k=1}^{n} \dfrac{1}{k} \approx \ln n + 0.5772$; the harmonic series diverges	Volume 1, Chapter 14
p-series	$\sum_{n=1}^{\infty} \dfrac{1}{n^p}$ converges iff $p > 1$; $\sum \dfrac{1}{n^2} = \dfrac{\pi^2}{6}$	Volume 1, Chapter 14

Fact	Formula	Ch.
Sigma rules	$\sum(a_k + b_k) = \sum a_k + \sum b_k$, $\sum c\,a_k = c\sum a_k$, $\sum_{k=1}^{n} c = nc$	Volume 1, Chapter 14
Recurrence (linear, first order)	$a_n = ra_{n-1} + d$ has fixed point $L = \dfrac{d}{1-r}$ and $a_n = L + r^n(a_0 - L)$ $(r \neq 1)$	Volume 1, Chapter 14
Taylor polynomial	$T_n(x) = \sum_{k=0}^{n} \dfrac{f^{(k)}(a)}{k!}(x-a)^k$	Chapter 18
Standard series	$e^x = \sum_{n\geq 0} \dfrac{x^n}{n!}$; $\sin x = x - \dfrac{x^3}{3!} + \dfrac{x^5}{5!} - \cdots$; $\cos x = 1 - \dfrac{x^2}{2!} + \dfrac{x^4}{4!} - \cdots$ (all x)	Chapter 18
More standard series	$\dfrac{1}{1-x} = \sum_{n\geq 0} x^n$ $(\lvert x\rvert < 1)$; $\ln(1+x) = x - \dfrac{x^2}{2} + \dfrac{x^3}{3} - \cdots$ $(-1 < x \leq 1)$	Chapter 18

A.9. Complex numbers

Fact	Formula	Ch.
Imaginary unit	$i^2 = -1$; powers cycle $i, -1, -i, 1$	Volume 1, Chapter 15
Arithmetic	$(a+bi) \pm (c+di) = (a \pm c) + (b \pm d)i$; $(a+bi)(c+di) = (ac - bd) + (ad + bc)i$	Volume 1, Chapter 15
Conjugate	$\overline{a+bi} = a - bi$; $z\bar{z} = a^2 + b^2$; $\overline{zw} = \bar{z}\,\bar{w}$	Volume 1, Chapter 15
Division	$\dfrac{a+bi}{c+di} = \dfrac{(a+bi)(c-di)}{c^2+d^2} = \dfrac{(ac+bd)+(bc-ad)i}{c^2+d^2}$	Volume 1, Chapter 15
Modulus and argument	$\lvert z\rvert = \sqrt{a^2+b^2}$; $\arg z = \operatorname{atan2}(b, a)$	Volume 1, Chapter 15
Polar / exponential form	$z = r(\cos\theta + i\sin\theta) = re^{i\theta}$	Volume 1, Chapter 15
Euler's formula	$e^{i\theta} = \cos\theta + i\sin\theta$; hence $e^{i\pi} + 1 = 0$	Volume 1, Chapter 15
Multiplying in polar form	Moduli multiply, arguments add: $r_1 e^{i\theta_1} \cdot r_2 e^{i\theta_2} = r_1 r_2\, e^{i(\theta_1+\theta_2)}$	Volume 1, Chapter 15

Fact	Formula	Ch.
De Moivre	$(\cos\theta + i\sin\theta)^n = \cos n\theta + i\sin n\theta$	Volume 1, Chapter 15
Roots of unity	Solutions of $z^n = 1$: $z_k = e^{2\pi ik/n}$, $k = 0, 1, \ldots, n-1$	Volume 1, Chapter 15
Real polynomials	Non-real roots come in conjugate pairs $a \pm bi$	Volume 1, Chapter 15

A.10. Limits and continuity

Fact	Formula	Ch.
Limit laws	Limits of sums, differences, products, constant multiples and quotients are the sums (etc.) of the limits, provided those exist and the denominator's limit is nonzero	Chapter 16
Continuity at a	$\lim_{x \to a} f(x) = f(a)$	Chapter 16
Formal definition	$\lim_{x \to a} f(x) = L$ means: for every $\varepsilon > 0$ there is $\delta > 0$ with $0 < \lvert x - a \rvert < \delta \implies \lvert f(x) - L \rvert < \varepsilon$	Chapter 16
Key trigonometric limits	$\lim_{x \to 0} \dfrac{\sin x}{x} = 1$, $\lim_{x \to 0} \dfrac{1 - \cos x}{x} = 0$	Chapter 16
Limits defining e	$\lim_{n \to \infty} \left(1 + \dfrac{1}{n}\right)^n = e$, $\lim_{x \to 0} \dfrac{e^x - 1}{x} = 1$	Chapter 16
Growth hierarchy as $x \to \infty$	$\ln x \ll x^p \ll a^x \ll x!$ (any $p > 0$, $a > 1$); e.g. $\dfrac{x^{100}}{e^x} \to 0$	Chapter 16
Rational functions at ∞	Compare degrees: top lower $\to 0$; equal $\to$ ratio of leading coefficients; top higher $\to \pm\infty$	Chapter 16
Squeeze theorem	If $g \le f \le h$ near a and $g, h \to L$, then $f \to L$	Chapter 16
L'Hôpital's rule	For $\frac{0}{0}$ or $\frac{\infty}{\infty}$ forms: $\lim \dfrac{f(x)}{g(x)} = \lim \dfrac{f'(x)}{g'(x)}$ if the right side exists	Chapter 18
Intermediate value theorem	If f is continuous on $[a, b]$ and y lies between $f(a)$ and $f(b)$, some c in $[a, b]$ has $f(c) = y$	Chapter 16

A.11. Derivatives

Rule	Formula	Ch.
Definition	$f'(x) = \lim_{h \to 0} \dfrac{f(x+h) - f(x)}{h}$	Chapter 17
Constant / linearity	$(c)' = 0$; $(af + bg)' = af' + bg'$	Chapter 17
Power rule	$\dfrac{d}{dx} x^n = n x^{n-1}$ (any real n)	Chapter 17
Product rule	$(fg)' = f'g + fg'$	Chapter 17
Quotient rule	$\left(\dfrac{f}{g}\right)' = \dfrac{f'g - fg'}{g^2}$	Chapter 17
Chain rule	$\dfrac{d}{dx} f(g(x)) = f'(g(x))\, g'(x)$; in Leibniz form $\dfrac{dy}{dx} = \dfrac{dy}{du}\dfrac{du}{dx}$	Chapter 17
Exponentials	$(e^x)' = e^x$; $(a^x)' = a^x \ln a$	Chapter 17
Logarithms	$(\ln x)' = \dfrac{1}{x}$; $(\log_a x)' = \dfrac{1}{x \ln a}$; $(\ln \lvert x \rvert)' = \dfrac{1}{x}$	Chapter 17
Trigonometric	$(\sin x)' = \cos x$; $(\cos x)' = -\sin x$; $(\tan x)' = \sec^2 x$	Chapter 17
Inverse trigonometric	$(\arcsin x)' = \dfrac{1}{\sqrt{1-x^2}}$; $(\arccos x)' = -\dfrac{1}{\sqrt{1-x^2}}$; $(\arctan x)' = \dfrac{1}{1+x^2}$	Chapter 17
Logistic sigmoid	$\sigma(x) = \dfrac{1}{1+e^{-x}}$, $\sigma'(x) = \sigma(x)\,(1-\sigma(x))$	Volume 3, Chapter 33
Implicit differentiation	Differentiate both sides with respect to x, treating y as $y(x)$: e.g. $x^2 + y^2 = 1 \implies 2x + 2yy' = 0 \implies y' = -x/y$	Chapter 17

Rule	Formula	Ch.
Tangent line / linearisation	$y = f(a) + f'(a)(x - a)$; so $f(a + h) \approx f(a) + hf'(a)$	Chapter 18
Critical points and tests	Critical point: $f'(c) = 0$ or undefined. Second-derivative test: $f''(c) > 0$ local min, $f''(c) < 0$ local max, $= 0$ inconclusive	Chapter 18
Closed-interval extrema	On $[a, b]$ check critical points and both endpoints; the largest/smallest value wins	Chapter 18
Concavity / inflection	$f'' > 0$ concave up, $f'' < 0$ concave down; inflection where f'' changes sign	Chapter 18
Mean value theorem	f continuous on $[a, b]$, differentiable inside $\implies f'(c) = \dfrac{f(b) - f(a)}{b - a}$ for some $c \in (a, b)$	Chapter 18
Newton's method	$x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$	Chapter 18

A.12. Integrals

Rule	Formula	Ch.
Power rule	$\int x^n\,dx = \dfrac{x^{n+1}}{n+1} + C\,(n \neq -1)$	Chapter 19
Reciprocal	$\int \dfrac{1}{x}\,dx = \ln\lvert x\rvert + C$	Chapter 19
Exponentials	$\int e^x\,dx = e^x + C$; $\int e^{kx}\,dx = \dfrac{1}{k}e^{kx} + C$; $\int a^x\,dx = \dfrac{a^x}{\ln a} + C$	Chapter 19
Trigonometric	$\int \sin x\,dx = -\cos x + C$; $\int \cos x\,dx = \sin x + C$; $\int \sec^2 x\,dx = \tan x + C$	Chapter 19
Inverse-trig forms	$\int \dfrac{dx}{1+x^2} = \arctan x + C$; $\int \dfrac{dx}{\sqrt{1-x^2}} = \arcsin x + C$	Chapter 19
Logarithm	$\int \ln x\,dx = x\ln x - x + C$	Chapter 19
Linearity	$\int (af + bg)\,dx = a\int f\,dx + b\int g\,dx$	Chapter 19
Fundamental theorem (part 1)	$\dfrac{d}{dx}\int_a^x f(t)\,dt = f(x)$	Chapter 19
Fundamental theorem (part 2)	$\int_a^b f(x)\,dx = F(b) - F(a)$ where $F' = f$	Chapter 19
Definite-integral properties	$\int_a^a f = 0$; $\int_b^a f = -\int_a^b f$; $\int_a^b f + \int_b^c f = \int_a^c f$	Chapter 19
Substitution	$\int f(g(x))\,g'(x)\,dx = \int f(u)\,du$ with $u = g(x)$; for definite integrals change the limits to $g(a), g(b)$	Chapter 19
Integration by parts	$\int u\,dv = uv - \int v\,du$	Chapter 19
Riemann sum	$\int_a^b f(x)\,dx = \lim_{n\to\infty}\sum_{i=1}^{n} f(x_i^*)\,\Delta x$, $\Delta x = \dfrac{b-a}{n}$	Chapter 19

Rule	Formula	Ch.
Area between curves	$\int_a^b (f(x) - g(x))dx$ where $f \geq g$ on $[a, b]$	Chapter 19
Average value	$\bar{f} = \frac{1}{b-a}\int_a^b f(x)\,dx$	Chapter 19
Volume of revolution (discs)	$V = \pi \int_a^b f(x)^2\,dx$ about the x-axis	Chapter 19
Gaussian integral	$\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$	Volume 3, Chapter 30

A.13. Multivariable calculus

Fact	Formula	Ch.
Partial derivative	$\dfrac{\partial f}{\partial x}$: differentiate in x holding the other variables fixed. Example: $f = x^2 y \implies f_x = 2xy,\ f_y = x^2$	Chapter 20
Gradient	$\nabla f = \left(\dfrac{\partial f}{\partial x_1}, \dots, \dfrac{\partial f}{\partial x_n} \right)$; points in the direction of steepest ascent; its length is that steepest rate	Chapter 20
Directional derivative	$D_{\mathbf{u}} f = \nabla f \cdot \mathbf{u}$ for a unit vector $\mathbf{u}$	Chapter 20
Gradient descent	$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k)$ (η the learning rate / step size)	Chapter 20
Tangent plane	$z = f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$	Chapter 20
Multivariable chain rule	$\dfrac{d}{dt} f(x(t), y(t)) = f_x \dfrac{dx}{dt} + f_y \dfrac{dy}{dt} = \nabla f \cdot \mathbf{r}'(t)$	Chapter 20
Critical points and second-derivative test	$\nabla f = \mathbf{0}$; with $D = f_{xx} f_{yy} - f_{xy}^2$: $D > 0, f_{xx} > 0$ min; $D > 0, f_{xx} < 0$ max; $D < 0$ saddle; $D = 0$ inconclusive	Chapter 20
Hessian	$H_{ij} = \dfrac{\partial^2 f}{\partial x_i \partial x_j}$; symmetric for smooth f	Chapter 20
Lagrange multipliers	Optimise f subject to $g = c$: solve $\nabla f = \lambda \nabla g,\ g = c$	Chapter 20

A.14. Logic

Law	Equivalence	Ch.
Conditional as "or"	$p \to q \equiv \neg p \lor q$; false only when p is true and q false	Volume 3, Chapter 21
Contrapositive (equivalent)	$p \to q \equiv \neg q \to \neg p$	Volume 3, Chapter 21
Converse and inverse (not equivalent to $p \to q$)	Converse $q \to p$; inverse $\neg p \to \neg q$ (these two are equivalent to each other)	Volume 3, Chapter 21
Negating a conditional	$\neg(p \to q) \equiv p \land \neg q$	Volume 3, Chapter 21
Biconditional	$p \leftrightarrow q \equiv (p \to q) \land (q \to p)$	Volume 3, Chapter 21
De Morgan	$\neg(p \land q) \equiv \neg p \lor \neg q$; $\neg(p \lor q) \equiv \neg p \land \neg q$	Volume 3, Chapter 21
Double negation	$\neg\neg p \equiv p$	Volume 3, Chapter 21
Distributive	$p \land (q \lor r) \equiv (p \land q) \lor (p \land r)$; $p \lor (q \land r) \equiv (p \lor q) \land (p \lor r)$	Volume 3, Chapter 21

Law	Equivalence	Ch.
Absorption	$p \vee (p \wedge q) \equiv p$; $p \wedge (p \vee q) \equiv p$	Volume 3, Chapter 21
Excluded middle / contradiction	$p \vee \neg p \equiv \mathbf{T}$; $p \wedge \neg p \equiv \mathbf{F}$	Volume 3, Chapter 21
Exclusive or	$p \oplus q \equiv (p \vee q) \wedge \neg(p \wedge q)$	Volume 3, Chapter 21
Negating quantifiers	$\neg \forall x\, P(x) \equiv \exists x\, \neg P(x)$; $\neg \exists x\, P(x) \equiv \forall x\, \neg P(x)$	Volume 3, Chapter 21
Quantifier order matters	$\forall x\, \exists y\, P(x, y)$ is generally *not* equivalent to $\exists y\, \forall x\, P(x, y)$	Volume 3, Chapter 21

A.15. Sets

Identity	Formula	Ch.
Basic operations	$A \cup B$ (in either), $A \cap B$ (in both), $A \setminus B$ (in A not B), A^c (not in A)	Volume 3, Chapter 21
Difference via complement	$A \setminus B = A \cap B^c$	Volume 3, Chapter 21
De Morgan for sets	$(A \cup B)^c = A^c \cap B^c$; $(A \cap B)^c = A^c \cup B^c$	Volume 3, Chapter 21
Distributive	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$; $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	Volume 3, Chapter 21
Subset tests	$A \subseteq B \iff A \cap B = A \iff A \cup B = B$	Volume 3, Chapter 21
Inclusion-exclusion (two sets)	$\lvert A \cup B \rvert = \lvert A \rvert + \lvert B \rvert - \lvert A \cap B \rvert$	Volume 3, Chapter 24
Inclusion-exclusion (three sets)	$\lvert A \cup B \cup C \rvert = \lvert A \rvert + \lvert B \rvert + \lvert C \rvert - \lvert A \cap B \rvert - \lvert A \cap C \rvert - \lvert B \cap C \rvert + \lvert A \cap B \cap C \rvert$	Volume 3, Chapter 24
Power set	$\lvert \mathcal{P}(A) \rvert = 2^{\lvert A \rvert}$	Volume 3, Chapter 21

Identity	Formula	Ch.
Cartesian product	$\lvert A \times B\rvert = \lvert A\rvert \cdot \lvert B\rvert$	Volume 3, Chapter 21

A.16. Proof techniques

Technique	To prove … do this	Ch.
Direct proof	$p \to q$: assume p, deduce q step by step	Volume 3, Chapter 22
Contrapositive	$p \to q$: assume $\neg q$, deduce $\neg p$	Volume 3, Chapter 22
Contradiction	p: assume $\neg p$, derive something impossible	Volume 3, Chapter 22
Cases	Split into exhaustive cases and prove each	Volume 3, Chapter 22
Counterexample	To *disprove* $\forall x\, P(x)$, exhibit one x with $\neg P(x)$	Volume 3, Chapter 22
Induction	$P(n)$ for all $n \geq n_0$: prove $P(n_0)$ (base case), then $P(k) \implies P(k+1)$ (inductive step)	Volume 3, Chapter 22
Strong induction	Inductive step assumes $P(n_0), \ldots, P(k)$ all hold, then proves $P(k+1)$	Volume 3, Chapter 22
Loop invariant	Induction applied to code: true before the loop, preserved by each iteration, so true at exit	Volume 3, Chapter 22

A.17. Counting

Situation	Count	Ch.
Sum rule	Choose one of m options *or* one of n (disjoint): $m + n$	Volume 3, Chapter 24
Product rule	Make a choice with m options *then* one with n: mn	Volume 3, Chapter 24
Ordered, repetition allowed	Sequences of length r from n symbols: n^r	Volume 3, Chapter 24
Arrangements of n distinct items	$n! = n(n-1)\cdots 2 \cdot 1$, with $0! = 1$	Volume 3, Chapter 24
Permutations (ordered, no repetition)	$P(n, r) = \dfrac{n!}{(n-r)!}$	Volume 3, Chapter 24
Combinations (unordered, no repetition)	$\dbinom{n}{r} = \dfrac{n!}{r!\,(n-r)!}$	Volume 3, Chapter 24
Multisets / stars and bars (unordered, repetition allowed)	Choose r from n types: $\dbinom{n+r-1}{r}$	Volume 3, Chapter 24
Arrangements with repeated items	$\dfrac{n!}{n_1!\,n_2!\cdots n_k!}$ (e.g. MISSISSIPPI: $\frac{11!}{1!\,4!\,4!\,2!} = 34650$)	Volume 3, Chapter 24

Situation	Count	Ch.
Pascal's rule and symmetry	$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$; $\binom{n}{k} = \binom{n}{n-k}$	Volume 3, Chapter 24
Row sum	$\sum_{k=0}^{n} \binom{n}{k} = 2^n$ (the number of subsets of an n-set)	Volume 3, Chapter 24
Derangements	Permutations with no fixed point: $D_n = n!\sum_{k=0}^{n} \frac{(-1)^k}{k!} \approx \frac{n!}{e}$	Volume 3, Chapter 24
Pigeonhole principle	n items in $m < n$ boxes: some box holds at least $\lceil n/m \rceil$ items	Volume 3, Chapter 24

A.18. Graphs

Fact	Formula	Ch.
Handshake lemma	$\sum_v \deg(v) = 2\lvert E\rvert$; so the number of odd-degree vertices is even	Volume 3, Chapter 25
Complete graph	K_n has $\binom{n}{2} = \frac{n(n-1)}{2}$ edges	Volume 3, Chapter 25
Trees	A tree on n vertices is connected, acyclic, and has exactly $n - 1$ edges	Volume 3, Chapter 25
Euler circuit	Exists in a connected graph iff every vertex has even degree (path: exactly 0 or 2 odd vertices)	Volume 3, Chapter 25
Euler's formula (planar, connected)	$V - E + F = 2$	Volume 3, Chapter 25
Traversal cost	BFS and DFS run in $O(V + E)$ with adjacency lists	Volume 3, Chapter 25

A.19. Modular arithmetic and number theory

Fact	Formula	Ch.
Congruence	$a \equiv b \pmod{m} \iff m \mid (a-b)$ (same remainder on division by m)	Volume 3, Chapter 23
Arithmetic respects congruence	If $a \equiv b$ and $c \equiv d \pmod{m}$ then $a \pm c \equiv b \pm d$ and $ac \equiv bd \pmod{m}$	Volume 3, Chapter 23
Reduce early	$(a \cdot b) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$; same for $+$	Volume 3, Chapter 23
Euclidean algorithm	$\gcd(a,b) = \gcd(b, a \bmod b)$, $\gcd(a,0) = a$	Volume 3, Chapter 23
Bézout's identity	$\gcd(a,b) = ax + by$ for some integers x, y (found by the extended Euclidean algorithm)	Volume 3, Chapter 23
Modular inverse	$a^{-1} \bmod m$ exists iff $\gcd(a,m) = 1$	Volume 3, Chapter 23
Fundamental theorem of arithmetic	Every integer $n \geq 2$ factors into primes uniquely (up to order)	Volume 3, Chapter 23
Fermat's little theorem	p prime, $p \nmid a \implies a^{p-1} \equiv 1 \pmod{p}$	Volume 3, Chapter 23

Fact	Formula	Ch.
Euler’s totient	$\varphi(n)$ = number of $1 \le k \le n$ coprime to n; $\varphi(p) = p-1$; $\varphi(pq) = (p-1)(q-1)$ for distinct primes	Volume 3, Chapter 23
Euler’s theorem	$\gcd(a,n)=1 \implies a^{\varphi(n)} \equiv 1 \pmod{n}$	Volume 3, Chapter 23
Chinese remainder theorem	If $m_1, \dots, m_k$ are pairwise coprime, the system $x \equiv a_i \pmod{m_i}$ has a unique solution modulo $m_1 m_2 \cdots m_k$	Volume 3, Chapter 23
RSA	$n = pq$; pick e coprime to $\varphi(n)$; $d \equiv e^{-1} \pmod{\varphi(n)}$. Encrypt $c = m^e \bmod n$; decrypt $m = c^d \bmod n$	Volume 3, Chapter 23
Fast exponentiation	Square-and-multiply computes $a^k \bmod m$ in $O(\log k)$ multiplications	Volume 3, Chapter 23

A.20. Vectors and matrices

Fact	Formula	Ch.
Length (norm)	$\|\mathbf{v}\| = \sqrt{v_1^2 + \cdots + v_n^2}$; unit vector $\hat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$	Chapter 26
Dot product	$\mathbf{u}\cdot\mathbf{v} = \sum_i u_i v_i = \|\mathbf{u}\|\|\mathbf{v}\|\cos\theta$	Chapter 26
Orthogonality	$\mathbf{u}\perp\mathbf{v} \iff \mathbf{u}\cdot\mathbf{v} = 0$	Chapter 26
Cosine similarity	$\cos\theta = \dfrac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$	Chapter 26
Projection of $\mathbf{u}$ onto $\mathbf{v}$	$\operatorname{proj}_{\mathbf{v}}\mathbf{u} = \dfrac{\mathbf{u}\cdot\mathbf{v}}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v}$	Chapter 26
Cross product (3D)	$\mathbf{u}\times\mathbf{v} = (u_2v_3 - u_3v_2,\ u_3v_1 - u_1v_3,\ u_1v_2 - u_2v_1)$; length $\|\mathbf{u}\|\|\mathbf{v}\|\sin\theta$, perpendicular to both	Chapter 26
Triangle and Cauchy-Schwarz inequalities	$\|\mathbf{u}+\mathbf{v}\| \le \|\mathbf{u}\| + \|\mathbf{v}\|$; $\|\mathbf{u}\cdot\mathbf{v}\| \le \|\mathbf{u}\|\|\mathbf{v}\|$	Chapter 26
Matrix product	$(AB)_{ij} = \sum_k A_{ik}B_{kj}$; needs (columns of A) = (rows of B); in general $AB \neq BA$	Chapter 27
Transpose rules	$(A^T)_{ij} = A_{ji}$; $(AB)^T = B^T A^T$	Chapter 27
Inverse rules	$AA^{-1} = A^{-1}A = I$; $(AB)^{-1} = B^{-1}A^{-1}$; $(A^T)^{-1} = (A^{-1})^T$	Chapter 27
2×2 determinant	$\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$	Chapter 27
2×2 inverse	$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \dfrac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$ (if $ad - bc \neq 0$)	Chapter 27
3×3 determinant (first-row expansion)	$a(ei - fh) - b(di - fg) + c(dh - eg)$ for rows $(a,b,c), (d,e,f), (g,h,i)$	Chapter 27
Determinant facts	$\det(AB) = \det A \det B$; $\det A^T = \det A$; A invertible $\iff \det A \neq 0$; $\lvert\det A\rvert$ is the area/volume scale factor	Chapter 27

Fact	Formula	Ch.
Rotation by θ (2D)	$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$; $R_\theta^{-1} = R_\theta^T = R_{-\theta}$	Chapter 27
Least squares (normal equations)	Minimise $\|A\mathbf{x} - \mathbf{b}\|$: solve $A^T A\mathbf{x} = A^T\mathbf{b}$	Chapter 27
Eigenvalues and eigenvectors	$A\mathbf{v} = \lambda\mathbf{v}$, $\mathbf{v} \neq \mathbf{0}$; eigenvalues solve $\det(A - \lambda I) = 0$	Chapter 28
2×2 characteristic equation	$\lambda^2 - (\operatorname{tr} A)\lambda + \det A = 0$; in general $\operatorname{tr} A = \sum \lambda_i$, $\det A = \prod \lambda_i$	Chapter 28
Diagonalisation / spectral theorem	$A = PDP^{-1}$; for symmetric A, $A = Q\Lambda Q^T$ with Q orthogonal and real eigenvalues	Chapter 28
Singular value decomposition	$A = U\Sigma V^T$ (U, V orthogonal, Σ diagonal, non-negative)	Chapter 28

A.21. Probability

Rule	Formula	Ch.
Range and total	$0 \le P(A) \le 1$, $P(\Omega) = 1$	Volume 3, Chapter 29
Equally likely outcomes	$P(A) = \dfrac{\text{number of outcomes in } A}{\text{total number of outcomes}}$	Volume 3, Chapter 29
Complement	$P(A^c) = 1 - P(A)$; "at least one" $= 1 - P(\text{none})$	Volume 3, Chapter 29
Addition rule	$P(A \cup B) = P(A) + P(B) - P(A \cap B)$; just $P(A) + P(B)$ if mutually exclusive	Volume 3, Chapter 29
Conditional probability	$P(A \mid B) = \dfrac{P(A \cap B)}{P(B)}$ $(P(B) > 0)$	Volume 3, Chapter 29
Multiplication rule	$P(A \cap B) = P(A \mid B)\,P(B) = P(B \mid A)\,P(A)$	Volume 3, Chapter 29
Independence	$P(A \cap B) = P(A)\,P(B)$, equivalently $P(A \mid B) = P(A)$	Volume 3, Chapter 29
Law of total probability	$P(A) = \sum_i P(A \mid B_i)\,P(B_i)$ for a partition $B_1, B_2, \ldots$	Volume 3, Chapter 29

Rule	Formula	Ch.
Bayes' theorem	$P(H \mid E) = \frac{P(E \mid H)\,P(H)}{P(E)} = \frac{P(E \mid H)\,P(H)}{P(E \mid H)\,P(H) + P(E \mid H^c)\,P(H^c)}$	Volume 3, Chapter 29
Birthday bound	With N equally likely values and n draws, $P(\text{some repeat}) \approx 1 - e^{-n(n-1)/(2N)}$; likely once $n \approx 1.18\sqrt{N}$	Volume 3, Chapter 29
Expectation	$E[X] = \sum_x x\,P(X = x)$ (discrete) or $\int x\,f(x)\,dx$ (continuous)	Volume 3, Chapter 30
Linearity of expectation	$E[aX + bY + c] = aE[X] + bE[Y] + c$ (always, even if dependent)	Volume 3, Chapter 30
Variance	$\mathrm{Var}(X) = E[(X - \mu)^2] = E[X^2] - (E[X])^2$; $\sigma = \sqrt{\mathrm{Var}(X)}$	Volume 3, Chapter 30
Variance rules	$\mathrm{Var}(aX + b) = a^2\,\mathrm{Var}(X)$; $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\,\mathrm{Cov}(X, Y)$	Volume 3, Chapter 30
Covariance and correlation	$\mathrm{Cov}(X, Y) = E[XY] - E[X]E[Y]$; $\rho = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y} \in [-1, 1]$; independent $\Longrightarrow$ covariance 0 (not conversely)	Volume 3, Chapter 30

A.22. Common distributions

Distribution	PMF or PDF	Mean	Variance	Ch.
Bernoulli (p)	$P(X=1)=p$, $P(X=0)=1-p$	p	$p(1-p)$	Volume 3, Chapter 30
Binomial (n,p)	$\binom{n}{k}p^k(1-p)^{n-k}$, $k=0,\dots,n$	np	$np(1-p)$	Volume 3, Chapter 30
Geometric (p) (trials until first success)	$(1-p)^{k-1}p$, $k=1,2,\dots$	$\frac{1}{p}$	$\frac{1-p}{p^2}$	Volume 3, Chapter 30
Poisson (λ)	$\frac{e^{-\lambda}\lambda^k}{k!}$, $k=0,1,2,\dots$	λ	λ	Volume 3, Chapter 30
Discrete uniform on $\{1,\dots,n\}$	$\frac{1}{n}$	$\frac{n+1}{2}$	$\frac{n^2-1}{12}$	Volume 3, Chapter 30
Continuous uniform on $[a,b]$	$\frac{1}{b-a}$ for $a\le x\le b$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$	Volume 3, Chapter 30
Exponential (λ)	$\lambda e^{-\lambda x}$ for $x\ge 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$	Volume 3, Chapter 30
Normal (μ,σ^2)	$\frac{1}{\sigma\sqrt{2\pi}}\,e^{-(x-\mu)^2/(2\sigma^2)}$	μ	σ^2	Volume 3, Chapter 30

Fact	Formula	Ch.
Standardising	$Z = \frac{X - \mu}{\sigma}$ has mean 0 and standard deviation 1	Volume 3, Chapter 30
68-95-99.7 rule	For a normal distribution, about 68%, 95% and 99.7% of values lie within 1, 2 and 3 standard deviations of the mean	Volume 3, Chapter 30
Central limit theorem	For n independent draws with mean μ and variance σ^2, the sample mean $\bar{X}$ is approximately $N(\mu, \sigma^2/n)$ for large n	Volume 3, Chapter 30
Law of large numbers	$\bar{X} \to \mu$ as $n \to \infty$	Volume 3, Chapter 30

A.23. Statistics

Quantity	Formula	Ch.
Mean	$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$	Volume 3, Chapter 31
Median, quartiles, IQR	Median: middle value of the sorted data (average the two middle values if n is even); $\text{IQR} = Q_3 - Q_1$; outlier rule of thumb: beyond $1.5 \times \text{IQR}$ from the quartiles	Volume 3, Chapter 31
Population variance	$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$	Volume 3, Chapter 31
Sample variance	$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$ (divide by $n-1$, Bessel's correction)	Volume 3, Chapter 31
z-score	$z = \frac{x - \bar{x}}{s}$ (or $\frac{x-\mu}{\sigma}$)	Volume 3, Chapter 31
Standard error of the mean	$\text{SE} = \frac{\sigma}{\sqrt{n}} \approx \frac{s}{\sqrt{n}}$	Volume 3, Chapter 31
Confidence interval for a mean	$\bar{x} \pm t^* \frac{s}{\sqrt{n}}$ ($t^* \approx z^* = 1.96$ for 95% with large n)	Volume 3, Chapter 31
Confidence interval for a proportion	$\hat{p} \pm z^* \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$	Volume 3, Chapter 31

Quantity	Formula	Ch.
Sample size for margin of error E	Proportion: $n \approx \left(\frac{z^*}{E}\right)^2 \hat{p}(1-\hat{p})$ (use $\hat{p} = 0.5$ if unknown); 95%, $\pm 3\%$ needs about 1068	Volume 3, Chapter 31
Correlation coefficient	$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \in [-1, 1]$	Volume 3, Chapter 31
Least-squares regression line	$\hat{y} = a + bx$ with $b = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = r\frac{s_y}{s_x}$, $a = \bar{y} - b\bar{x}$	Volume 3, Chapter 31
Coefficient of determination	$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$; equals r^2 for a straight-line fit	Volume 3, Chapter 31
Hypothesis test	p-value = probability, if the null hypothesis were true, of data at least as extreme as observed. Reject H_0 if p-value $< \alpha$ (often 0.05)	Volume 3, Chapter 31
Test statistic for a mean	$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$ ($n - 1$ degrees of freedom)	Volume 3, Chapter 31

A.24. Numerical methods

Method	Formula	Ch.
Newton's method	$x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$; quadratic convergence near a simple root (correct digits roughly double each step); can diverge from a poor start or where $f' \approx 0$	Chapter 32
Square root by Newton (Heron)	$x_{n+1} = \frac{1}{2}\left(x_n + \dfrac{a}{x_n}\right) \to \sqrt{a}$	Chapter 18
Bisection	If $f(a)$ and $f(b)$ differ in sign, test the midpoint $m = \frac{a+b}{2}$ and keep the half where the sign changes. After n halvings the bracket has width $(b-a)/2^n$; need about $\log_2\left((b-a)/\text{tol}\right)$ steps	Chapter 32
Secant method	$x_{n+1} = x_n - f(x_n)\dfrac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$ (Newton without a derivative)	Chapter 32
Fixed-point iteration	$x_{n+1} = g(x_n)$ converges near a fixed point x^* if $\lvert g'(x^*)\rvert < 1$	Chapter 32
Finite differences	Forward $f'(x) \approx \dfrac{f(x+h) - f(x)}{h}$ (error $O(h)$); central $f'(x) \approx \dfrac{f(x+h) - f(x-h)}{2h}$ (error $O(h^2)$)	Chapter 32
Trapezoid rule	$\int_a^b f \approx \dfrac{h}{2}(f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-1} + f_n)$, $h = \dfrac{b-a}{n}$, $f_i = f(a+ih)$; error $O(h^2)$	Chapter 32
Simpson's rule (n even)	$\int_a^b f \approx \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{n-2} + 4f_{n-1} + f_n)$; error $O(h^4)$	Chapter 32
Euler's method (ODE $y' = f(t, y)$)	$y_{n+1} = y_n + h\, f(t_n, y_n)$; error $O(h)$	Chapter 32
Floating point (IEEE double)	Machine epsilon $2^{-52} \approx 2.2 \times 10^{-16}$; about 15-16 significant decimal digits; never test floats with $==$, use a tolerance	Chapter 32
Absolute and relative error	$\lvert\hat{x} - x\rvert$ and $\dfrac{\lvert\hat{x} - x\rvert}{\lvert x\rvert}$	Chapter 32

A.25. Optimisation, machine learning, and information

Quantity	Formula	Ch.
Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	Volume 3, Chapter 33
Softmax	$\mathrm{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$ (subtract $\max_j z_j$ first for numerical safety)	Volume 3, Chapter 33
Entropy (bits)	$H(p) = -\sum_i p_i \log_2 p_i$; maximal ($\log_2 n$) for the uniform distribution on n outcomes	Volume 3, Chapter 33
Cross-entropy	$H(p, q) = -\sum_i p_i \log q_i$; with a one-hot label it is $-\log q_{\text{true}}$	Volume 3, Chapter 33
Kullback-Leibler divergence	$D_{\mathrm{KL}}(p \,\|\, q) = \sum_i p_i \log \frac{p_i}{q_i} = H(p, q) - H(p) \geq 0$	Volume 3, Chapter 33

Appendix B: Glossary

Every Refresher box in Volume 2, alphabetised. This is the book's reference mode: the same definitions you met in context, collected for lookup. Each entry links back to the chapter where it is taught.

B.1. A

Absolute and relative error (Chapter 32)

If $\hat{x}$ approximates x, the **absolute error** is $|\hat{x} - x|$ and the **relative error** is $|\hat{x} - x|/|x|$ (for $x \neq 0$). Relative error 10^{-k} means roughly k correct **significant digits**. Floating point controls relative error; that is its whole design.

Accumulation (Chapter 19)

An **accumulated quantity** is a total built up from a rate over time (or over any other variable). Distance accumulates from speed, water in a tank accumulates from flow rate, money accumulates from income per month. If the rate is constant, total = rate × duration. If the rate varies, total = area under the rate graph. The **integral** is the tool for computing that area.

$\mathbb{R}^2$, $\mathbb{R}^n$, and bold vectors (Chapter 20)

$\mathbb{R}$ is the set of real numbers. $\mathbb{R}^2$ is the set of all ordered pairs (x, y) of real numbers, i.e. the plane. $\mathbb{R}^n$ is the set of all ordered lists of n real numbers, $(x_1, x_2, \ldots, x_n)$.

So $f : \mathbb{R}^2 \to \mathbb{R}$ reads "f takes a pair of reals to a real".

When there are many inputs we bundle them into one object written in **bold**: $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, and write $f(\mathbf{x})$ instead of $f(x_1, \ldots, x_n)$. In handwriting, bold becomes an underline or an arrow: $\vec{x}$. A model with a million weights is just a function $L : \mathbb{R}^{1\,000\,000} \to \mathbb{R}$.

Antiderivative (Chapter 19)

An **antiderivative** of f is a function F with $F'(x) = f(x)$. For example, $F(x) = x^3/3$ is an antiderivative of $f(x) = x^2$, because $\frac{d}{dx}(x^3/3) = x^2$. So is $x^3/3 + 7$: antiderivatives are only determined up to an added constant.

Average value (Chapter 19)

The **average value** of f on $[a, b]$ is

$$\bar{f} = \frac{1}{b-a} \int_a^b f(x)\,dx.$$

Geometrically, it is the height of the rectangle on $[a, b]$ with the same area as the region under f.

B.2. B

Big-O notation for errors (Chapter 32)

$E(h) = O(h^p)$ as $h \to 0$ means $|E(h)| \leq Ch^p$ for some constant C and all small enough h. A method with error $O(h^p)$ has **order** p: halving h divides the error by about 2^p. This is the same symbol as in algorithm analysis, but there $n \to \infty$; here $h \to 0$.

B.3. C

Characteristic polynomial, trace (Chapter 28)

- The **characteristic polynomial** of an $n \times n$ matrix is $\det(A - \lambda I)$, a polynomial of degree n in λ. Its roots are the eigenvalues; setting it to zero gives the **characteristic equation**.
- The **trace** $\operatorname{tr} A$ is the sum of the diagonal entries. It equals the sum of the eigenvalues; the determinant equals their product.
- The **eigenspace** of λ is the set of all solutions of $(A - \lambda I)\mathbf{v} = \mathbf{0}$: all eigenvectors for λ together with $\mathbf{0}$.
- The set of all eigenvalues is the **spectrum** of A.

Choosing between eigendecomposition, PCA and SVD (Chapter 28)

- **Square matrix applied repeatedly** (dynamics, Markov chains, recurrences, PageRank): eigenvalues and eigenvectors. The largest eigenvalue governs the long run.
- **Symmetric matrix** (covariance, Laplacian, Hessian): eigendecomposition with `eigh`; orthogonal eigenvectors guaranteed.
- **Data cloud, directions of most variation:** PCA, computed via the SVD of the centred data.
- **Any rectangular matrix, compression, denoising, rank, least squares:** the SVD.

Component (Chapter 26)

The **components** of a vector are the numbers in its list, one per axis. The vector $(3, 4)$ has **x-component** 3 and **y-component** 4. In general the i-th component of $\mathbf{v}$ is written v_i, so $\mathbf{v} = (v_1, v_2, \ldots, v_n)$. The set of all lists of n real numbers is written $\mathbb{R}^n$ ("R-n"), so $(3, 4) \in \mathbb{R}^2$ means "(3, 4) is a vector in two-dimensional real space".

Concave up, concave down, inflection point (Chapter 18)

f is **concave up** (UK: **convex**) on an interval where its slope is increasing — the graph bends like a cup, ∪. It is **concave down** (UK: **concave**) where its slope is decreasing — a cap, ∩. Test: $f'' > 0 \Rightarrow$ concave up; $f'' < 0 \Rightarrow$ concave down. An **inflection point** is a point on the graph where the concavity changes.

Condition number (Chapter 32)

The **condition number** κ of a problem measures how much relative changes in the input are amplified in the output:

$$\frac{\text{relative change in output}}{\text{relative change in input}} \approx \kappa.$$

For a differentiable function of one variable, $\kappa(x) = \left| \dfrac{x\,f'(x)}{f(x)} \right|$. For a linear system $A\mathbf{x} = \mathbf{b}$, $\kappa(A) = \|A\|\,\|A^{-1}\|$ (Chapter 27). A problem is **well-conditioned** if κ is modest and **ill-conditioned** if it is large. For the matrix above, $\kappa \approx 4 \times 10^4$.

Continuity at a point (Chapter 16)

A function f is **continuous at** a if all three conditions hold:

1. $f(a)$ is defined (a is in the domain);
2. $\lim_{x \to a} f(x)$ exists;
3. the two are equal: $\lim_{x \to a} f(x) = f(a)$.

f is **continuous on an interval** if it is continuous at every point of it (using one-sided limits at closed endpoints), and simply **continuous** if it is continuous at every point of its domain. A point where continuity fails is a **discontinuity**.

Covariance matrix, principal components (Chapter 28)

- The **covariance matrix** of centred data X (rows = observations) is $C = \frac{1}{N-1} X^{\mathsf{T}} X$. Entry c_{jk} is the covariance of features j and k; the diagonal holds variances. It's symmetric with non-negative eigenvalues.
- The **principal components** (PCs) are the eigenvectors of C, sorted by decreasing eigenvalue: orthogonal directions of decreasing variance.
- The **scores** of a data point are its coordinates along the PCs: XQ_k for the top k components. Keeping $k < p$ of them is **dimensionality reduction**.

Critical point, saddle point, Hessian (Chapter 20)

A **critical point** (or **stationary point**) of f is a point where $\nabla f = \mathbf{0}$, i.e. every first partial derivative is zero.

A **saddle point** is a critical point that is neither a local maximum nor a local minimum: f rises in some directions and falls in others.

The **Hessian** is the matrix of second partial derivatives:

$$H = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}.$$

It is symmetric ($f_{xy} = f_{yx}$) and plays the role of the second derivative: it measures curvature.

Critical point (stationary point) (Chapter 18)

A **critical point** of f is a point c in its domain where $f'(c) = 0$ or $f'(c)$ does not exist. Where $f'(c) = 0$ specifically, UK texts say **stationary point**: the tangent is horizontal. Every local maximum or minimum of a function at an interior point is a critical point — but not every critical point is a maximum or minimum.

Cross product (Chapter 26)

The **cross product** (vector product) $\mathbf{a} \times \mathbf{b}$ of two 3D vectors is a 3D vector perpendicular to both, of length $\|\mathbf{a}\| \, \|\mathbf{b}\| \sin\theta$, oriented by the right-hand rule. Contrast the **dot** product, which works in any dimension and returns a **scalar** involving $\cos\theta$. In unit-vector form: $\mathbf{i} \times \mathbf{j} = \mathbf{k}$, $\mathbf{j} \times \mathbf{k} = \mathbf{i}$, $\mathbf{k} \times \mathbf{i} = \mathbf{j}$.

B.4. D

Definite integral (Chapter 19)

The **definite integral** $\int_a^b f(x)\,dx$ is a **number**: the limit of Riemann sums of f over $[a, b]$. Geometrically it is the **signed area** between the graph of f and the x-axis from a to b. A function for which this limit exists is called **integrable** on $[a, b]$.

Derivative, difference quotient, differentiable (Chapter 17)

The **difference quotient** is $\dfrac{f(x+h) - f(x)}{h}$, the slope of a secant.

The **derivative** $f'(x)$ is the limit of the difference quotient as $h \to 0$: the slope of the tangent, or the instantaneous rate of change of f.

A function is **differentiable at** x if that limit exists (as a finite number). It is **differentiable** (on an interval) if it is differentiable at every point there.

Differentiation is the process of finding a derivative. The word comes from Leibniz's "differences", the tiny changes dx and dy.

Determinant (Chapter 27)

The **determinant** of a square matrix, written $\det A$ or $|A|$, is the signed factor by which the transformation scales area (2D) or volume (3D and up).

- $|\det A|$ = scale factor for area/volume.
- $\det A < 0$: orientation is reversed (a mirror image).
- $\det A = 0$: space is flattened into a lower dimension; the matrix is **singular**.
- For 2×2: $\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$.
- $\det(AB) = \det A \cdot \det B$ (scale factors multiply when you compose) and $\det I = 1$.

Diagonalisable, similar, change of basis (Chapter 28)

- A square matrix is **diagonalisable** if $A = PDP^{-1}$ with D diagonal. This works exactly when A has n linearly independent eigenvectors (the columns of P); the eigenvalues fill the diagonal of D in the same order.
- A **change-of-basis matrix** P has the new basis vectors as columns; $P\mathbf{c}$ converts new coordinates to standard ones, $P^{-1}\mathbf{x}$ converts back.

- Matrices A and $B = P^{-1}AP$ are **similar**: the same transformation described in different coordinates. Similar matrices share eigenvalues, trace and determinant.
- n **distinct** eigenvalues guarantee diagonalisability. Repeated eigenvalues may or may not allow it ($2I$ yes, the shear no).

Differential equation (Chapter 19)

A **differential equation** is an equation involving an unknown function and its derivatives, such as $\frac{dy}{dt} = ky$. A **solution** is a function that makes the equation true. An **initial condition** such as $y(0) = y_0$ picks out one solution from the family. An **ordinary differential equation** (ODE) has a single independent variable (usually time).

Differential equation, initial value problem (Chapter 32)

A **differential equation** relates an unknown function to its derivatives. An **initial value problem** (IVP) gives $\mathbf{y}'(t) = F(t, \mathbf{y})$ together with a starting value $\mathbf{y}(t_0) = \mathbf{y}_0$. Any higher-order equation becomes a first-order **system** by adding variables: for the pendulum take the state $\mathbf{y} = (\theta, \omega)$ with angular velocity $\omega = \theta'$, so $\theta' = \omega,\ \omega' = -\frac{g}{L}\sin\theta$. The **step size** h is the time increment between computed states.

Directional derivative (Chapter 20)

The **directional derivative** $D_{\mathbf{u}}f(\mathbf{a})$ is the rate of change of f at $\mathbf{a}$ per unit distance moved in the direction of the **unit** vector $\mathbf{u}$. It equals $\nabla f(\mathbf{a}) \cdot \mathbf{u}$.

Dot product (Chapter 20)

The **dot product** of two vectors multiplies matching components and adds the results:

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 \ (+ \cdots + a_n b_n).$$

It also has a geometric meaning, which Chapter 26 derives from the law of cosines:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}|\,|\mathbf{b}| \cos\theta,$$

where θ is the angle between the arrows. So the dot product is largest when the vectors point the same way ($\cos 0 = 1$), zero when they are perpendicular ($\cos 90^\circ = 0$), and most negative when they point opposite ways.

Dot product (Chapter 26)

The **dot product** (scalar product, inner product) of two vectors of the same dimension is

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = \|\mathbf{a}\|\,\|\mathbf{b}\| \cos\theta.$$

It is a scalar. Other notations: $\langle \mathbf{a}, \mathbf{b} \rangle$ (mathematics, physics), $\mathbf{a}^{\mathsf{T}}\mathbf{b}$ (linear algebra, "a-transpose b"; see Chapter 27), and `a @ b` or `np.dot(a, b)` in NumPy.

Double integral, iterated integral, dA (Chapter 20)

The **double integral** $\iint_R f\,dA$ is the signed volume between the surface $z = f(x, y)$ and the region R of the plane. dA stands for a tiny element of area, $dx\,dy$ in Cartesian coordinates and $r\,dr\,d\theta$ in polar coordinates (the r is the Jacobian determinant from earlier).

An **iterated integral** evaluates it as two ordinary integrals, inner first. While doing the inner integral, the outer variable is held constant, exactly as with partial derivatives.

$[a, b] \times [c, d]$ (read "a to b cross c to d") is the rectangle of points with $a \le x \le b$ and $c \le y \le d$.

Dummy variable (Chapter 19)

The variable of integration is a **dummy variable**: its name does not matter. $\int_0^2 x^2\,dx$, $\int_0^2 t^2\,dt$ and $\int_0^2 u^2\,du$ are all the same number, $8/3$. It is exactly like a loop variable or a bound variable in a lambda: `sum(i*i for i in r)` and `sum(k*k for k in r)` are the same.

B.5. E

Eigenvector and eigenvalue (Chapter 28)

A nonzero vector $\mathbf{v}$ is an **eigenvector** of a square matrix A if

$$A\mathbf{v} = \lambda\mathbf{v}$$

for some scalar λ (Greek "lambda"), called the **eigenvalue** belonging to $\mathbf{v}$. In words: A maps $\mathbf{v}$ to a multiple of itself. $\lambda > 1$ stretches, $0 < \lambda < 1$ shrinks, $\lambda < 0$ flips (and scales by $|\lambda|$), $\lambda = 0$ crushes $\mathbf{v}$ to zero. Any nonzero multiple of an eigenvector is also an eigenvector with the same eigenvalue, so really an eigenvector names a **direction** (a line through the origin). The zero vector is never counted as an eigenvector, since $A\mathbf{0} = \lambda\mathbf{0}$ for every λ and tells you nothing.

B.6. F

Function of several variables (Chapter 20)

A **function of several variables** (or **multivariable function**) takes more than one input number and returns one output number. For example, $f(x, y) = x^2 + 3y^2$ takes two inputs, and $T(x, y, z)$ could give the temperature at a point in a room.

A **scalar-valued** function returns a single number. (A function returning several numbers, such as a position in space, is **vector-valued**. We'll meet one when we discuss the Jacobian.)

Fundamental Theorem of Calculus, Part 1 (Chapter 19)

If f is continuous on $[a, b]$ and $A(x) = \int_a^x f(t)\,dt$, then A is differentiable and

$$\frac{d}{dx}\int_a^x f(t)\,dt = f(x).$$

In words: **the rate of change of accumulated area is the current height**. Differentiation undoes integration.

Fundamental Theorem of Calculus, Part 2 (the evaluation theorem) (Chapter 19)

If f is continuous on $[a, b]$ and F is any antiderivative of f, then

$$\int_a^b f(x)\,dx = F(b) - F(a) = \Big[F(x)\Big]_a^b.$$

The bracket notation $\big[F(x)\big]_a^b$ (also written $F(x)\big|_a^b$) means "evaluate at the top limit, minus evaluate at the bottom limit".

B.7. G

Gradient and the symbol ∇ (Chapter 20)

The **gradient** of $f(x_1, \ldots, x_n)$ is the vector of its partial derivatives:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right).$$

The symbol ∇ is called **nabla** (or "del"). Read $\nabla f(\mathbf{x})$ as "the gradient of f at the point $\mathbf{x}$". It is a vector with one component for each input, and it can be different at every point. In some texts it is also written $\mathbf{grad}\, f$ or $\frac{\partial f}{\partial \mathbf{x}}$.

B.8. H

Higher derivatives (Chapter 17)

The **second derivative** is the derivative of the derivative: $f''(x)$, or $\dfrac{d^2y}{dx^2}$ ("dee two y by dee x squared"), or $\ddot{s}$ in mechanics. The Leibniz form comes from applying the operator twice: $\frac{d}{dx}\left(\frac{dy}{dx}\right)$.

The nth derivative is written $f^{(n)}(x)$ (parentheses, so it is not mistaken for a power) or $\dfrac{d^n y}{dx^n}$.

In mechanics: **position** s, **velocity** $v = \dot{s} = \frac{ds}{dt}$, **acceleration** $a = \dot{v} = \ddot{s} = \frac{d^2s}{dt^2}$. The third derivative is called **jerk**.

Homogeneous coordinates (Chapter 27)

Homogeneous coordinates represent a 2D point (x, y) as $(x, y, 1)$ and a 3D point (x, y, z) as $(x, y, z, 1)$, so that translations, rotations, scalings and (in 3D graphics) perspective projections are all matrix multiplications: 3×3 matrices for 2D graphics, 4×4 for 3D. Directions use a final coordinate of 0. More generally (X, Y, W) with $W \neq 0$ represents the point $(X/W, Y/W)$; dividing by W is what makes perspective work.

B.9. I

Identity matrix and matrix product (Chapter 27)

- The **identity matrix** I_n is the $n \times n$ matrix with 1s on the main diagonal and 0s elsewhere. $I\mathbf{x} = \mathbf{x}$ for every $\mathbf{x}$.

- The **product** AB is defined when (columns of A) = (rows of B), and $(AB)_{ij}$ = (row i of A) · (column j of B). It represents applying B then A.
- Laws: associative $(AB)C = A(BC)$; distributive $A(B + C) = AB + AC$; **not** commutative in general.

Implicit and explicit (Chapter 17)

A function is given **explicitly** when y is isolated: $y = \sqrt{25 - x^2}$. It is given **implicitly** by an equation relating x and y without isolating either: $x^2 + y^2 = 25$. **Implicit differentiation** finds dy/dx directly from the implicit equation.

Improper integral (Chapter 19)

An **improper integral** is one where the interval is unbounded (a limit of integration is $\pm\infty$) or the integrand blows up somewhere in the interval. It is defined as a limit of ordinary integrals and is said to **converge** if that limit is a finite number and **diverge** otherwise.

Increasing and decreasing (Chapter 18)

A function f is **increasing** on an interval if $a < b$ implies $f(a) < f(b)$ for any two points in the interval, and **decreasing** if $a < b$ implies $f(a) > f(b)$. Test: $f' > 0$ on an interval ⇒ increasing there; $f' < 0$ ⇒ decreasing.

Indefinite integral (Chapter 19)

The **indefinite integral** $\int f(x)\,dx$ (no limits) denotes the **family of all antiderivatives** of f:

$$\int f(x)\,dx = F(x) + C, \qquad \text{where } F'(x) = f(x).$$

C is the **constant of integration**: an arbitrary constant, because adding a constant does not change a derivative. A definite integral is a number; an indefinite integral is a family of functions.

Indeterminate form (Chapter 16)

An **indeterminate form** is a limit whose naive substitution gives an expression like $\frac{0}{0}$, $\frac{\infty}{\infty}$, $\infty - \infty$, $0 \cdot \infty$, or 1^∞. "Indeterminate" means the form alone does not determine the answer: $\frac{x}{x}$, $\frac{x^2}{x}$, and $\frac{x}{x^2}$ are all $\frac{0}{0}$ at 0, with limits 1, 0, and "does not exist". The expression must be transformed until substitution works.

Indeterminate form (Chapter 18)

An **indeterminate form** is a limit shape whose value cannot be decided from the limits of the pieces alone: $\frac{0}{0}$, $\frac{\infty}{\infty}$, $0 \cdot \infty$, $\infty - \infty$, 0^0, 1^∞, ∞^0. L'Hôpital applies directly only to the first two; the others are first rewritten as a quotient.

Infinite limit (Chapter 16)

$\lim_{x \to a} f(x) = \infty$ means $f(x)$ becomes larger than any given number when x is close enough to a ($x \neq a$). Similarly for $-\infty$. The limit does **not** exist as a real number; the notation records the direction of failure. The line $x = a$ is then a **vertical asymptote** of the graph.

Integration by parts (Chapter 19)

Integration by parts is the formula $\int u\,dv = uv - \int v\,du$, obtained by integrating the product rule. For definite integrals: $\int_a^b u\,dv = \big[uv\big]_a^b - \int_a^b v\,du$. A rough guide for choosing u is **LIATE**: Logarithmic, Inverse trig, Algebraic (powers of x), Trigonometric, Exponential, taking as u whichever comes first in that list.

Intermediate value theorem (Chapter 16)

Intermediate value theorem (IVT): if f is continuous on the closed interval $[a, b]$ and N is any number between $f(a)$ and $f(b)$, then there is at least one c in $[a, b]$ with $f(c) = N$.

The most-used special case: if f is continuous on $[a, b]$ and $f(a)$ and $f(b)$ have **opposite signs**, then f has a **root** (a zero) somewhere in (a, b).

Invertible and singular (Chapter 27)

A square matrix A is **invertible** (or **non-singular**) if there's a matrix A^{-1} with $AA^{-1} = A^{-1}A = I$. Otherwise it is **singular**. For a square matrix, all of these are equivalent: A is invertible; $\det A \neq 0$; the columns are linearly independent; the columns span the whole space; $A\mathbf{x} = \mathbf{b}$ has exactly one solution for every $\mathbf{b}$; $A\mathbf{x} = \mathbf{0}$ only for $\mathbf{x} = \mathbf{0}$. Non-square matrices never have a two-sided inverse.

B.10. J

Jacobian matrix (Chapter 20)

The **Jacobian** of $\mathbf{F}(x, y) = \big(F_1(x, y), F_2(x, y)\big)$ is the matrix of all first partial derivatives:

$$J = \begin{bmatrix} \dfrac{\partial F_1}{\partial x} & \dfrac{\partial F_1}{\partial y} \\[2ex] \dfrac{\partial F_2}{\partial x} & \dfrac{\partial F_2}{\partial y} \end{bmatrix}.$$

Row i is ∇F_i. The Jacobian is the multi-output version of the derivative: small input changes $\Delta\mathbf{x}$ produce output changes $\Delta\mathbf{F} \approx J\,\Delta\mathbf{x}$. Matrix–vector multiplication is explained in Chapter 27.

B.11. L

Lagrange multiplier (Chapter 20)

To optimise $f(\mathbf{x})$ subject to a constraint $g(\mathbf{x}) = c$, look for points where $\nabla f = \lambda \nabla g$ and $g = c$. The number λ (Greek "lambda") is the **Lagrange multiplier**. It measures how fast the optimal value of f changes as the constraint level c is relaxed: $\lambda = df^*/dc$. Economists call it the **shadow price** of the constraint.

Law of cosines (Chapter 26)

In any triangle with sides a, b, c, where θ is the angle between sides a and b (opposite side c):

$$c^2 = a^2 + b^2 - 2ab\cos\theta.$$

When $\theta = 90°$, $\cos\theta = 0$ and it reduces to Pythagoras. The $-2ab\cos\theta$ term is the correction for the triangle not being right-angled. (Derived in Volume 1, Chapter 13 .)

Learning rate, iterate, convergence (Chapter 20)

The **learning rate** (or **step size**) η is the positive number multiplying the gradient in each update. Machine learning calls it the learning rate. Numerical analysis calls it the step size and often writes α.

Each $\mathbf{x}_k$ is an **iterate**. The method **converges** if the iterates approach a limit (ideally a minimum), and **diverges** if they run off, typically growing without bound.

Limit at infinity (Chapter 16)

$\lim_{x\to\infty} f(x) = L$ means $f(x)$ can be made as close to L as we like by taking x sufficiently large. Similarly for $x \to -\infty$. If such an L exists, the horizontal line $y = L$ is a **horizontal asymptote**. For a sequence (Volume 1, Chapter 14), $\lim_{n\to\infty} a_n = L$ means the same thing with n restricted to whole numbers; this is exactly the "convergence" of the previous chapter, now with a name.

Limit (formal, ε–δ) (Chapter 16)

$\lim_{x\to a} f(x) = L$ means: for every $\varepsilon > 0$ there exists a $\delta > 0$ such that

$$0 < |x - a| < \delta \quad \Longrightarrow \quad |f(x) - L| < \varepsilon.$$

Here $|x - a|$ is the distance between x and a (Volume 1, Chapter 5); "$0 < |x - a|$" excludes $x = a$ itself; and $\Longrightarrow$ reads "implies". In the language of logic (Volume 3, Chapter 21):
$\forall \varepsilon > 0\ \exists \delta > 0\ \forall x\ (0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon)$.

Limit (informal) (Chapter 16)

We write

$$\lim_{x\to a} f(x) = L$$

and say "the **limit** of $f(x)$ as x **approaches** (or **tends to**) a is L" if $f(x)$ can be made as close to L as we like by taking x sufficiently close to a, *but not equal to* a. Reading the notation piece by piece: **lim** abbreviates "limit"; the subscript $x \to a$ says which variable is moving and where it is heading; $f(x)$ is the expression being watched; L is the value it approaches. An alternative notation is $f(x) \to L$ as $x \to a$.

Linear combination, span, independence, basis (Chapter 26)

- A **linear combination** of $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is any vector $c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_k\mathbf{v}_k$ with scalar **coefficients** c_i.
- The **span** of a set of vectors is the set of all their linear combinations.
- Vectors are **linearly independent** if none of them is a linear combination of the others: each contributes a genuinely new direction. Otherwise they're **linearly dependent**.

- A **basis** of a space is a linearly independent set that spans it. Every vector in the space is then a combination of basis vectors in exactly one way. The number of basis vectors is the **dimension** of the space.
- The **standard basis** of $\mathbb{R}^n$ is $\mathbf{e}_1, \ldots, \mathbf{e}_n$, and the components of a vector are precisely its coefficients in that basis: $(3, 4) = 3\mathbf{e}_1 + 4\mathbf{e}_2$.

Linear transformation (Chapter 27)

A **linear transformation** (or **linear map**) $T : \mathbb{R}^n \to \mathbb{R}^m$ is a function from n-dimensional vectors to m-dimensional vectors satisfying $T(\mathbf{u} + \mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$ and $T(c\mathbf{v}) = cT(\mathbf{v})$. The arrow notation $T : \mathbb{R}^n \to \mathbb{R}^m$ reads "T maps R-n to R-m": the **domain** is $\mathbb{R}^n$, the **codomain** $\mathbb{R}^m$. Geometrically: the origin is fixed, and grid lines stay straight, parallel and evenly spaced.

Linearisation, differential (Chapter 18)

The **linearisation** (or **tangent line approximation**) of f at a is $L(x) = f(a) + f'(a)(x - a)$, and $f(x) \approx L(x)$ for x near a.

The **differentials** dx and dy restate this: dx is a small change in the input, and $dy = f'(x)\,dx$ is the corresponding change *along the tangent line*. The true change is $\Delta y = f(x + dx) - f(x)$, and $\Delta y \approx dy$. This is where Leibniz's notation $\frac{dy}{dx}$ earns its look: $dy = \frac{dy}{dx}\,dx$.

Linearisation, tangent plane, differential (Chapter 20)

The **linearisation** (or **linear approximation**) of f at $\mathbf{a}$ is $L(\mathbf{x}) = f(\mathbf{a}) + \nabla f(\mathbf{a}) \cdot (\mathbf{x} - \mathbf{a})$. Its graph, for two inputs, is the **tangent plane**.

The **total differential** writes the same idea in terms of small changes: $df = f_x\,dx + f_y\,dy$.

Local and global extrema (Chapter 18)

f has a **local maximum** at c if $f(c) \geq f(x)$ for all x *near* c; a **local minimum** if $f(c) \leq f(x)$ nearby. A **global** (or **absolute**) maximum or minimum compares against *every* point of the domain. **Extremum** (plural **extrema**) means "maximum or minimum". UK texts often say **turning point** for a local extremum.

B.12. M

Machine epsilon and unit roundoff (Chapter 32)

Machine epsilon $\varepsilon_{\text{mach}}$ is the gap between 1 and the next larger floating-point number. For doubles $\varepsilon_{\text{mach}} = 2^{-52} \approx 2.22 \times 10^{-16}$. The **unit roundoff** $u = \varepsilon_{\text{mach}}/2 = 2^{-53} \approx 1.11 \times 10^{-16}$ is the largest relative error committed by rounding a real number (in range) to the nearest double. "Doubles carry about 16 significant decimal digits" is shorthand for $-\log_{10} u \approx 15.95$.

Magnitude (norm) and summation notation (Chapter 26)

The **magnitude**, **length** or **norm** of $\mathbf{v}$ is written $\|\mathbf{v}\|$ (double bars; some books use single bars $|\mathbf{v}|$). It's the **Euclidean norm**, also called the **L2 norm** and written $\|\mathbf{v}\|_2$ in machine-learning papers.

The symbol $\sum_{i=1}^{n} v_i^2$ is **summation notation**: Σ (capital Greek sigma, for "sum") means "add up", the **index** i runs from the value below (1) to the value above (n), and v_i^2 is the term being added for each i. So $\sum_{i=1}^{3} v_i^2 = v_1^2 + v_2^2 + v_3^2$. In code it's `sum(x*x for x in v)`.

Markov chain, transition matrix, steady state (Chapter 28)

A **Markov chain** is a random process that moves between **states**, where the probability of the next state depends only on the current one (it has no memory of earlier history). Its **transition matrix** P has entry p_{ij} = probability of moving **to** state i **from** state j. With this convention (**column-stochastic**), columns sum to 1 and distributions update as $\mathbf{x} \mapsto P\mathbf{x}$. Many statistics texts use the transpose (**row-stochastic**, rows sum to 1, $\mathbf{x}^{\mathsf{T}} \mapsto \mathbf{x}^{\mathsf{T}} P$); check which a source uses. A **steady state** (or **stationary distribution**) is a probability vector with $P\mathbf{x} = \mathbf{x}$: an eigenvector with eigenvalue 1, scaled so its entries sum to 1.

Matrix (Chapter 27)

A **matrix** is a rectangular array of numbers. An $m \times n$ matrix ("m by n") has m **rows** and n **columns**; rows first, always. The **entry** in row i, column j is written a_{ij} (so a_{12} is row 1, column 2). As a transformation, an $m \times n$ matrix takes n-dimensional vectors to m-dimensional vectors: column j is where the j-th basis vector lands. A **square** matrix has $m = n$ and maps a space to itself. The word is Latin for "womb" (the thing from which something grows), chosen by James Sylvester in 1850.

B.13. N

Non-square matrices change dimension (Chapter 27)

A 2×3 matrix has 3 columns (images of $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ in 3D), each with 2 entries (they land in 2D). So it maps $\mathbb{R}^3 \to \mathbb{R}^2$, for example by dropping the depth of a 3D point to draw it on paper: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ keeps x, y and discards z. An $m \times n$ matrix eats n-vectors and produces m-vectors. A neural-network layer from 768 to 3072 dimensions is a 3072×768 matrix.

Notations for the derivative (Chapter 17)

Lagrange (prime) notation: $f'(x)$, $f''(x)$ for the second derivative. Compact; good when the function has a name.

Leibniz notation: if $y = f(x)$, write $\frac{dy}{dx}$ ("dee y by dee x"). It looks like $\frac{\Delta y}{\Delta x}$ with the deltas made infinitely small, which is exactly the intuition. Value at a point: $\left.\frac{dy}{dx}\right|_{x=a}$.

Operator notation: $\frac{d}{dx}$ is an *operator*: a machine that takes a function and returns its derivative. $\frac{d}{dx}(x^2) = 2x$ reads "the derivative with respect to x of x^2 is $2x$". Also written D_x or D.

Newton (dot) notation: $\dot{x}$ for the derivative of x with respect to *time*, $\ddot{x}$ for the second. Common in mechanics and physics.

Numerical analysis (Chapter 32)

Numerical analysis is the branch of mathematics that designs and analyses algorithms for approximating answers to problems (roots, integrals, derivatives, solutions of equations) when exact symbolic answers are unavailable, too expensive, or meaningless because the data are themselves approximate. Its two central questions are *how fast does the method approach the answer?* and *how much error does it introduce?*

B.14. O

Objective function and constraint (Chapter 18)

In an optimisation problem, the **objective function** is the quantity you want to maximise or minimise (volume, area, cost, error). A **constraint** is a condition the variables must satisfy (fixed fence length, fixed volume). The **feasible set** or **domain** is the set of values satisfying all constraints. The best value of the objective is the **optimum**; the x achieving it is the **optimiser** ($\arg\max$ or $\arg\min$).

One-sided limits (Chapter 16)

$\lim_{x \to a^-} f(x) = L$ (the **left-hand limit**) means $f(x)$ approaches L as x approaches a *from below* ($x < a$). $\lim_{x \to a^+} f(x) = L$ (the **right-hand limit**) means the same with $x > a$. The superscript $-$ or $+$ is a direction, not a sign on a: $x \to 2^-$ means x increases towards 2 through values like 1.9, 1.99, …

Key fact: the two-sided limit $\lim_{x \to a} f(x)$ exists and equals L *exactly when* both one-sided limits exist and both equal L.

Order of convergence (Chapter 18)

A method has **order of convergence** p if $|e_{n+1}| \approx C|e_n|^p$ for large n. $p = 1$ is **linear convergence** (a fixed number of digits gained per step — bisection gains one binary digit); $p = 2$ is **quadratic** (digits double — Newton at a simple root). The secant method has $p = \frac{1+\sqrt{5}}{2} \approx 1.618$, the golden ratio.

Order of convergence (Chapter 32)

Let $e_n = |x_n - r|$ be the error after n steps. A method **converges with order** p if

$$e_{n+1} \approx C\, e_n^p$$

for some constant $C > 0$ as $n \to \infty$. **Linear** convergence ($p = 1$, need $C < 1$) adds a fixed number of correct digits per step. **Quadratic** convergence ($p = 2$) roughly *doubles* the number of correct digits per step. **Superlinear** means anything faster than linear.

Orthogonal (Chapter 26)

Two vectors are **orthogonal** if their dot product is zero. For nonzero vectors this means they meet at 90°. (The zero vector is orthogonal to everything, by this definition.) "Orthogonal" is the general-dimension word; "perpendicular" is usual in 2D and 3D geometry; "normal" is used for a vector

perpendicular to a surface. A set of vectors that are mutually orthogonal **and** all of length 1 is **orthonormal**; the standard basis is the classic example.

B.15. P

Partial derivative and the symbol ∂ (Chapter 20)

The **partial derivative** of f with respect to x is the rate of change of f as x varies with every other input held fixed.

Notations, all meaning the same thing:

$$\frac{\partial f}{\partial x}, \qquad f_x, \qquad \partial_x f, \qquad D_x f.$$

The curly ∂ is read "partial" (or "del", or "dee"). It replaces the straight d of $\frac{df}{dx}$ as a reminder that other variables exist and are being held fixed. To evaluate at a point, write $\frac{\partial f}{\partial x}(1,2)$ or $f_x(1,2)$.

Partition, subinterval, width (Chapter 19)

A **partition** of $[a, b]$ splits it into smaller **subintervals** using points $a = x_0 < x_1 < \cdots < x_n = b$. With equal widths, each subinterval has width

$$\Delta x = \frac{b-a}{n},$$

and the partition points are $x_i = a + i\,\Delta x$ for $i = 0, 1, \ldots, n$. The symbol Δ ("delta") means "a change in", so Δx is read "delta x": a small step in x.

Pivot (Chapter 32)

In Gaussian elimination the **pivot** is the entry used to eliminate the entries below it in the same column. **Partial pivoting** chooses, at each column, the row with the largest absolute value in that column (among the rows not yet used) and swaps it into the pivot position. **Back substitution** solves the resulting upper-triangular system from the last equation upward.

Projection (Chapter 26)

- **Scalar projection** of $\mathbf{a}$ onto $\mathbf{b}$: $\mathbf{a} \cdot \hat{\mathbf{b}} = \dfrac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|}$, a signed length.
- **Vector projection**: $\operatorname{proj}_{\mathbf{b}} \mathbf{a} = \dfrac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{b} \cdot \mathbf{b}}\,\mathbf{b}$, a vector along $\mathbf{b}$.
- **Orthogonal decomposition**: $\mathbf{a} = \operatorname{proj}_{\mathbf{b}} \mathbf{a} + (\mathbf{a} - \operatorname{proj}_{\mathbf{b}} \mathbf{a})$, one part parallel to $\mathbf{b}$, one perpendicular.

A key reading: **dotting with a unit vector measures the component in that direction.** That's why $\mathbf{v} \cdot \mathbf{i}$ is just v_1.

B.16. Q

Quadrature (Chapter 19)

Quadrature means numerical approximation of a definite integral. The name comes from the ancient problem of "squaring" a shape: constructing a square with the same area. A **quadrature rule** is a weighted sum $\sum w_i f(x_i)$ of function values at chosen **nodes** x_i.

B.17. R

Range, overflow, underflow (Chapter 32)

The largest finite double is about 1.80×10^{308}; the smallest positive normal double is about 2.23×10^{-308}; the smallest positive subnormal is about 4.9×10^{-324}. A result too large to represent **overflows** to $\pm\infty$. A result too small **underflows**, first gradually through the subnormals (losing precision) and finally to zero.

Rank (Chapter 27)

The **rank** of a matrix is the number of linearly independent columns (equivalently, of independent rows: the two counts always agree), which is the dimension of the set of all outputs $A\mathbf{x}$ (the **column space** or **image**). An $n \times n$ matrix is invertible exactly when its rank is n (**full rank**). The input directions squashed to zero form the **null space** (or **kernel**); its dimension is $n - \text{rank}$.

Rate of change and the symbol Δ (Chapter 17)

The Greek capital **delta**, Δ, means "change in". If a quantity goes from d_1 to d_2, then $\Delta d = d_2 - d_1$. It is a single symbol: Δt is not $\Delta \times t$.

The **average rate of change** of d with respect to t over an interval is $\dfrac{\Delta d}{\Delta t}$. For distance against time this is average speed; for any function it is the slope of the straight line joining the two endpoints of the graph.

Riemann sum (Chapter 19)

A **Riemann sum** for f on $[a, b]$ is any sum of the form $\sum_{i=1}^{n} f(x_i^*)\,\Delta x$: heights times widths, added. Each term is the (signed) area of one thin rectangle. Named after Bernhard Riemann (1826–1866), who made the definition precise.

Root, zero, iteration (Chapter 18)

A **root** or **zero** of f is a value r with $f(r) = 0$; graphically, where the curve crosses or touches the x-axis. An **iterative method** produces a sequence of approximations $x_0, x_1, x_2, \ldots$ (read "x-nought, x-one, ..."; the subscript is the step number), each computed from the previous one by the same rule. x_0 is the **initial guess**. The method **converges** if $x_n \to r$ as $n \to \infty$ (a limit of a sequence, Volume 1, Chapter 14).

Rules of differentiation (Chapter 17)

Rule	Formula
Constant	$\frac{d}{dx} c = 0$
Power	$\frac{d}{dx} x^n = n x^{n-1}$ (any real n)
Constant multiple	$(cf)' = cf'$
Sum / difference	$(f \pm g)' = f' \pm g'$
Product	$(fg)' = f'g + fg'$
Quotient	$(f/g)' = (f'g - fg')/g^2$
Chain	$\frac{d}{dx} f(g(x)) = f'(g(x))\, g'(x)$, or $\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$

B.18. S

Sample point (Chapter 19)

A **sample point** x_i^* (read "x i star") is the point inside the i-th subinterval where you measure the height $f(x_i^*)$. Left, right and midpoint sums are three particular choices of sample point.

Scalar and vector (Chapter 26)

A **scalar** is an ordinary single number: a size with no direction (mass 4 kg, temperature 20 °C, speed 30 m/s). A **vector** is a quantity with both **magnitude** (size) and **direction** (displacement 30 m north-east, velocity 30 m/s due south). In the list view, a vector in n dimensions is an ordered list of n real numbers. The word "scalar" comes from the fact that multiplying a vector by a number **scales** it.

Secant line and tangent line (Chapter 17)

A **secant line** (from Latin *secare*, to cut) is a line through two points on a curve. Its slope, $\dfrac{f(a+h) - f(a)}{h}$, is the average rate of change of f between a and $a + h$.

A **tangent line** (from Latin *tangere*, to touch) at a point P is the line through P whose direction matches the curve's direction at P. Zoom in far enough on a smooth curve at P and the curve and its tangent become indistinguishable.

Note: the tangent *line* is unrelated to the trig function $\tan$ except by shared etymology.

Second and mixed partial derivatives (Chapter 20)

Differentiating a partial derivative again gives a **second partial derivative**:

$$f_{xx} = \frac{\partial^2 f}{\partial x^2}, \qquad f_{yy} = \frac{\partial^2 f}{\partial y^2}, \qquad f_{xy} = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) = \frac{\partial^2 f}{\partial y\, \partial x}.$$

f_{xy} and f_{yx} are **mixed partials**. For every function you will meet in practice (any function whose second partials are continuous), **Clairaut's theorem** says they are equal: $f_{xy} = f_{yx}$. The order of differentiation does not matter.

Sigma notation (Chapter 19)

The capital Greek letter Σ (sigma, for "sum") abbreviates a long addition:

$$\sum_{i=1}^{n} f(x_i^*)\,\Delta x \;=\; f(x_1^*)\,\Delta x + f(x_2^*)\,\Delta x + \cdots + f(x_n^*)\,\Delta x.$$

Piece by piece: i is the **index** (a loop counter); $i = 1$ underneath is where it starts; n on top is where it stops (inclusive); the expression to the right is the **term** evaluated for each i and then added. In code it is `sum(f(x_star[i]) * dx for i in range(1, n + 1))`.

Significand, exponent, floating point (Chapter 32)

A **floating-point number** has the form $\pm m \times \beta^e$, where β is the **base** (2 in hardware), m is the **significand** (also called the **mantissa**) holding a fixed number of significant digits, and e is the integer **exponent**. In base 2 the significand of a nonzero number can always be scaled so that it looks like $1.f_1f_2f_3\ldots$ in binary; this is the **normalised** form, and the bits $f_1f_2\ldots$ are the **fraction**.

Singular value decomposition (Chapter 28)

Every real $m \times n$ matrix can be written $A = U\Sigma V^{\mathsf{T}}$ with U, V orthogonal and Σ diagonal with non-negative entries $\sigma_1 \geq \sigma_2 \geq \cdots$ (the **singular values**). The singular values are the square roots of the eigenvalues of the symmetric matrix $A^{\mathsf{T}}A$; the columns of V are its eigenvectors. The number of nonzero singular values is the **rank** of A. For a symmetric matrix with non-negative eigenvalues, the SVD and the eigendecomposition coincide.

Socks and shoes (Chapter 27)

The inverse of a product reverses the order: $(AB)^{-1} = B^{-1}A^{-1}$. To undo "socks, then shoes" you remove shoes first, then socks. Check: $(AB)(B^{-1}A^{-1}) = A(BB^{-1})A^{-1} = AIA^{-1} = AA^{-1} = I$. Also $(A^{-1})^{-1} = A$ and $\det(A^{-1}) = 1/\det A$ (if A multiplies areas by 3, undoing it divides them by 3).

The spectral theorem (Chapter 28)

If A is a real symmetric $n \times n$ matrix, then:

1. all its eigenvalues are real (no rotation hiding inside);
2. eigenvectors for different eigenvalues are orthogonal;
3. it always has n orthonormal eigenvectors, so it can be diagonalised as

$$A = Q\Lambda Q^{\mathsf{T}},$$

where Q is an orthogonal matrix (orthonormal eigenvector columns, so $Q^{-1} = Q^{\mathsf{T}}$) and Λ (capital lambda) is diagonal.

Geometrically: **every symmetric matrix is a pure stretch along some set of perpendicular axes**. Rotate to those axes (Q^{T}), stretch (Λ), rotate back (Q). No shear, no rotation.

Squeeze theorem (Chapter 16)

Squeeze theorem (also **sandwich** or **pinching theorem**): if $g(x) \le f(x) \le h(x)$ for all x near a (except possibly at a), and $\lim_{x \to a} g(x) = \lim_{x \to a} h(x) = L$, then $\lim_{x \to a} f(x) = L$ as well.

Standard derivatives (Chapter 17)

$f(x)$	$f'(x)$	$f(x)$	$f'(x)$
x^n	nx^{n-1}	e^x	e^x
$\sin x$	$\cos x$	b^x	$(\ln b)\, b^x$
$\cos x$	$-\sin x$	$\ln x$	$1/x$
$\tan x$	$\sec^2 x$	$\arctan x$	$1/(1+x^2)$

(Angles in radians. The $\arctan$ entry is derived in the next section.)

Surface, level curve, contour (Chapter 20)

The **graph** (or **surface**) of $f(x, y)$ is the set of points (x, y, z) with $z = f(x, y)$.

A **level curve** of f at level c is the set of points in the plane where $f(x, y) = c$. A **contour plot** draws several level curves at evenly spaced levels. With three inputs, the same idea gives **level surfaces** $f(x, y, z) = c$.

B.19. T

Taylor polynomial (Chapter 18)

The **degree-n Taylor polynomial** of f about a is $T_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k$, the unique polynomial of degree at most n matching f and its first n derivatives at a. Notation: $f^{(k)}$ is the k-th derivative (brackets distinguish it from a power), with $f^{(0)} = f$. Letting $n \to \infty$ gives the **Taylor series**.

TF-IDF and cosine similarity (Chapter 26)

- **Bag of words**: a document as a vector of word counts, ignoring order.
- **TF-IDF** (term frequency–inverse document frequency): each count multiplied by $\log(N/\text{df})$, so common words count for little and rare words for a lot.
- **Cosine similarity**: $\dfrac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\,\|\mathbf{b}\|}$, the cosine of the angle between two vectors; it ignores length. **Cosine distance** is $1 - \text{cosine similarity}$.

Transpose, symmetric, orthogonal (Chapter 27)

- **Transpose A^T**: rows and columns swapped. $(A^\mathsf{T})^\mathsf{T} = A$, $(AB)^\mathsf{T} = B^\mathsf{T}A^\mathsf{T}$, $\mathbf{a}\cdot\mathbf{b} = \mathbf{a}^\mathsf{T}\mathbf{b}$. Other notations: A^T, A' (statistics, MATLAB), `A.T` (NumPy).
- **Symmetric matrix**: $A^\mathsf{T} = A$ (a mirror image across the diagonal). Covariance matrices and distance matrices are symmetric; Chapter 28 shows they have especially beautiful eigenvectors.
- **Orthogonal matrix**: $Q^\mathsf{T}Q = I$, so $Q^{-1} = Q^\mathsf{T}$. Its columns are orthonormal; it preserves lengths and angles; $\det Q = \pm 1$.

B.20. U

Ulp (Chapter 32)

An **ulp** (unit in the last place) of a floating-point number x is the gap between x and the next representable number of larger magnitude. $\mathrm{ulp}(1) = 2^{-52}$; $\mathrm{ulp}(10^{16}) = 2$. Errors are often quoted in ulps: "correctly rounded" means within half an ulp.

u-substitution (Chapter 19)

Substitution (or **u-substitution**, or **change of variables**) rewrites an integral in terms of a new variable $u = g(x)$, replacing $g'(x)\,dx$ by du. It is the chain rule run backwards. It works when the integrand contains an inner function **and** (up to a constant factor) that inner function's derivative.

B.21. V

Vector arithmetic laws (Chapter 26)

For vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ in $\mathbb{R}^n$ and scalars s, t:

- $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$ (**commutative**)
- $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$ (**associative**)
- $\mathbf{a} + \mathbf{0} = \mathbf{a}$ and $\mathbf{a} + (-\mathbf{a}) = \mathbf{0}$
- $s(\mathbf{a} + \mathbf{b}) = s\mathbf{a} + s\mathbf{b}$ and $(s + t)\mathbf{a} = s\mathbf{a} + t\mathbf{a}$ (**distributive**)

Each follows immediately from the matching law for ordinary numbers, applied one component at a time. Any collection of objects obeying these rules is called a **vector space**; that abstraction is why polynomials, audio signals and images can all be treated as vectors.

Vector, length, unit vector (Chapter 20)

A **vector** in $\mathbb{R}^2$ is an ordered pair $\mathbf{v} = (v_1, v_2)$, pictured as an arrow from the origin to the point (v_1, v_2). It has a direction and a length. The same idea works with n components.

Its **length** (or **magnitude**, or **norm**) is $|\mathbf{v}| = \sqrt{v_1^2 + v_2^2}$ (Pythagoras; see Volume 1, Chapter 10).

A **unit vector** has length 1 and is used purely to indicate a direction. Any nonzero vector can be turned into one by dividing by its length: $\mathbf{u} = \mathbf{v}/|\mathbf{v}|$. For example, $(3, 4)$ has length 5, so $(0.6,\ 0.8)$ is the unit vector in that direction.

Vector notation (Chapter 26)

- **Bold lower-case** in print: $\mathbf{v}$, $\mathbf{a}$, $\mathbf{F}$ (force).
- **An arrow or an underline** by hand, since you can't write in bold: $\vec{v}$ or $\underline{v}$. UK A-level Mechanics used the underline.
- **A column vector**, the standard in linear algebra:

$$\mathbf{v} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}.$$

- **A row or tuple**: $(3, 4)$ or $\langle 3, 4 \rangle$ (the angle brackets are common in US calculus books, to distinguish a vector from a point).
- **Unit-vector form**: $3\mathbf{i} + 4\mathbf{j}$, where $\mathbf{i}$ and $\mathbf{j}$ are the arrows of length 1 along the x- and y-axes (and $\mathbf{k}$ along the z-axis in 3D). This is the form you'll remember from Mechanics.
- **An arrow between points**: $\overrightarrow{AB}$ is the vector from point A to point B.

In this book, vectors are bold. We write them as tuples in running text to save space and as columns inside calculations, especially once matrices arrive in Chapter 27.

Appendix C: Notation and Symbols

Mathematical notation is a compressed language. This appendix is its phrasebook: for each symbol, how to say it aloud, what it means, a tiny example, and the chapter where it is properly introduced. Saying a formula aloud is half of understanding it, so the "Read aloud" column matters more than it looks.

Where a symbol has more than one common meaning, all are listed; context decides. The section Section C.13 collects the worst offenders, and Section C.14 shows how to take a dense formula apart.

C.1. Arithmetic, comparison, and approximation

Symbol	Read aloud	Meaning	Example	Ch.
$=$	"equals"	Both sides name the same value	$2 + 3 = 5$	Volume 1, Chapter 1
$\neq$	"is not equal to"	The two sides differ	$0.1 + 0.2 \neq 0.3$ in floating point	Volume 1, Chapter 1
$<$, $>$	"is less than", "is greater than"	Strict order; the small end points at the smaller number	$-3 < 2$	Volume 1, Chapter 1
$\leq$, $\geq$	"is less than or equal to", "is greater than or equal to"	Order allowing equality	$x \geq 0$	Volume 1, Chapter 5
$\approx$	"is approximately equal to"	Close in value; how close depends on context	$\pi \approx 3.1416$	Volume 1, Chapter 1
$\equiv$	"is identically equal to"; "is congruent to"; "is logically equivalent to"	(1) true for every value of the variable; (2) same remainder modulo m; (3) same truth value in every case	$(x+1)^2 \equiv x^2 + 2x + 1$; $17 \equiv 2 \pmod 5$	Volume 1, Chapter 4, Volume 3, Chapter 23, Volume 3, Chapter 21

Symbol	Read aloud	Meaning	Example	Ch.
$:=$	"is defined to be"	Defines the left side; not a claim to check	$f(x) := x^2$	Volume 1, Chapter 4
$\propto$	"is proportional to"	Equal up to a constant multiplier	$A \propto r^2$ for circles (constant π)	Volume 1, Chapter 2
$\ll,\ \gg$	"is much less than", "is much greater than"	Smaller (larger) by orders of magnitude	$10^{-16} \ll 1$	Chapter 32
$\pm,\ \mp$	"plus or minus", "minus or plus"	Two cases at once; $\mp$ takes the opposite sign to a $\pm$ in the same formula	$x = 3 \pm 2$ means $x = 5$ or $x = 1$	Volume 1, Chapter 8
$\times,\ \cdot$	"times"	Multiplication (also cross product and dot product for vectors, Cartesian product for sets)	$3 \times 4 = 3 \cdot 4 = 12$	Volume 1, Chapter 1
$a/b,\ \frac{a}{b},\ a \div b$	"a over b", "a divided by b"	Division; a fraction	$\frac{6}{4} = 1.5$	Volume 1, Chapter 2
$a : b$	"a to b"	Ratio	$3 : 2$	Volume 1, Chapter 2
$\%$	"percent"	Per hundred: $p\% = p/100$	$15\% = 0.15$	Volume 1, Chapter 2
a^n	"a to the n", "a to the power n" (a^2 "a squared", a^3 "a cubed")	Repeated multiplication, extended to all real n	$2^{-1} = \frac{1}{2}$	Volume 1, Chapter 3

Symbol	Read aloud	Meaning	Example	Ch.
$\sqrt{a},\ \sqrt[n]{a}$	"root a", "the n-th root of a"	The non-negative number whose square (n-th power) is a	$\sqrt[3]{8} = 2$	Volume 1, Chapter 3
$\lvert x \rvert$	"the absolute value of x", "mod x"	Distance from 0	$\lvert -7 \rvert = 7$	Volume 1, Chapter 1
$\lfloor x \rfloor$	"floor of x"	Largest integer $\leq x$	$\lfloor 2.7 \rfloor = 2$, $\lfloor -2.7 \rfloor = -3$	Volume 1, Chapter 11
$\lceil x \rceil$	"ceiling of x"	Smallest integer $\geq x$	$\lceil 2.1 \rceil = 3$	Volume 1, Chapter 11
$n!$	"n factorial"	$1 \cdot 2 \cdots n$, with $0! = 1$	$4! = 24$	Volume 3, Chapter 24
∞	"infinity"	Not a number: shorthand for "without bound"	$x \to \infty$, $(0, \infty)$	Chapter 16
$\ldots,\ \cdots$	"dot dot dot", "and so on"	Continue the evident pattern	$1 + 2 + \cdots + n$	Volume 1, Chapter 14
$\therefore$	"therefore"	Introduces a conclusion	$2x = 6 \therefore x = 3$	Volume 3, Chapter 22
■ or □	"QED", "end of proof"	The proof is complete		Volume 3, Chapter 22

C.2. Number systems and intervals

Symbol	Read aloud	Meaning	Example	Ch.
$\mathbb{N}$	"the naturals", "N"	Natural numbers $\{0, 1, 2, \ldots\}$ (some authors start at 1)	$5 \in \mathbb{N}$	Volume 1, Chapter 1
$\mathbb{Z}$	"the integers", "Z" (from German *Zahlen*)	$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$	$-4 \in \mathbb{Z}$	Volume 1, Chapter 1
$\mathbb{Q}$	"the rationals", "Q" (for *quotient*)	Fractions p/q with p, q integers, $q \neq 0$	$0.75 = \frac{3}{4} \in \mathbb{Q}$	Volume 1, Chapter 1
$\mathbb{R}$	"the reals", "R"	All points on the number line	$\sqrt{2} \in \mathbb{R}$, $\sqrt{2} \notin \mathbb{Q}$	Volume 1, Chapter 1
$\mathbb{C}$	"the complex numbers", "C"	Numbers $a + bi$ with $a, b \in \mathbb{R}$	$3 - 2i \in \mathbb{C}$	Volume 1, Chapter 15
$\mathbb{R}^n$	"R n"	Lists (vectors) of n real numbers	$(1, -2, 0.5) \in \mathbb{R}^3$	Chapter 26
$\mathbb{Z}_m$ or $\mathbb{Z}/m\mathbb{Z}$	"the integers mod m"	The remainders $\{0, 1, \ldots, m-1\}$ with modular arithmetic	In $\mathbb{Z}_{12}$, $9 + 5 = 2$	Volume 3, Chapter 23

Table 2. Interval notation

Notation	Means x satisfies	Endpoints
(a, b)	$a < x < b$	both excluded ("open")
$[a, b]$	$a \le x \le b$	both included ("closed")
$[a, b)$	$a \le x < b$	left included, right excluded ("half-open"; Python's `range`)
$(a, b]$	$a < x \le b$	left excluded, right included
(a, ∞)	$x > a$	∞ is never included
$(-\infty, b]$	$x \le b$	
$(-\infty, \infty)$	any real x	the whole of $\mathbb{R}$

A round bracket means "up to but not including"; a square bracket means "including". Some European texts write $]a, b[$ for (a, b). Do not confuse the open interval (a, b) with the point (a, b) in the plane; context tells them apart. See Volume 1, Chapter 5 .

C.3. Sets

Symbol	Read aloud	Meaning	Example	Ch.
$\{\ \}$	"the set containing ..."	Unordered collection, no repeats	$\{1, 2, 3\} = \{3, 1, 2\}$	Volume 3, Chapter 21
$\{x \mid P(x)\}$ or $\{x : P(x)\}$	"the set of x such that P of x"	Set-builder notation: everything satisfying a condition	$\{x \in \mathbb{Z} \mid x^2 < 5\} = \{-2, \ldots, 2\}$	Volume 3, Chapter 21
$\in$	"is in", "is an element of", "belongs to"	Membership	$3 \in \{1, 2, 3\}$	Volume 3, Chapter 21
$\notin$	"is not in"	Non-membership	$4 \notin \{1, 2, 3\}$	Volume 3, Chapter 21
$\subseteq$	"is a subset of"	Every element of the left is in the right	$\{1, 2\} \subseteq \{1, 2, 3\}$	Volume 3, Chapter 21
$\subset$ or $\subsetneq$	"is a proper subset of"	Subset but not equal (some books use $\subset$ for $\subseteq$; check the convention)	$\mathbb{N} \subset \mathbb{Z}$	Volume 3, Chapter 21
$\cup$	"union", "A cup B"	In either set (or both)	$\{1, 2\} \cup \{2, 3\} = \{1, 2, 3\}$	Volume 3, Chapter 21
$\cap$	"intersection", "A cap B"	In both sets	$\{1, 2\} \cap \{2, 3\} = \{2\}$	Volume 3, Chapter 21

Symbol	Read aloud	Meaning	Example	Ch.
$A \setminus B$ or $A - B$	"A minus B", "A without B"	In A but not in B	$\{1,2,3\} \setminus \{2\} = \{1,3\}$	Volume 3, Chapter 21
A^c, $\overline{A}$, A'	"A complement"	Everything (in the universe) not in A	In $\{1,\ldots,6\}$, $\{1,2\}^c = \{3,4,5,6\}$	Volume 3, Chapter 21
$\emptyset$ or $\varnothing$	"the empty set"	The set with no elements	$\{1\} \cap \{2\} = \emptyset$	Volume 3, Chapter 21
$\lvert A \rvert$ or $\#A$	"the size of A", "the cardinality of A"	Number of elements	$\lvert\{a,b,c\}\rvert = 3$	Volume 3, Chapter 21
$A \times B$	"A cross B"	Cartesian product: all ordered pairs (a,b)	$\{1,2\} \times \{x\} = \{(1,x),(2,x)\}$	Volume 3, Chapter 21
$\mathcal{P}(A)$	"the power set of A"	The set of all subsets of A	$\mathcal{P}(\{1,2\}) = \{\emptyset,\{1\},\{2\},\{1,2\}\}$	Volume 3, Chapter 21

C.4. Logic

Symbol	Read aloud	Meaning	Example	Ch.
$\neg p$	"not p"	True exactly when p is false (code: `!p`, `not p`)	$\neg(2 > 3)$ is true	Volume 3, Chapter 21
$p \wedge q$	"p and q"	True when both are true (code: `&&`, `and`)	$x > 0 \wedge x < 1$	Volume 3, Chapter 21
$p \vee q$	"p or q"	True when at least one is true; inclusive or (code: `or`)	$x < 0 \vee x > 1$	Volume 3, Chapter 21
$p \oplus q$	"p xor q", "exclusive or"	True when exactly one is true (code: ^ on booleans)	$\mathbf{T} \oplus \mathbf{T}$ is false	Volume 3, Chapter 21
$p \to q$	"if p then q", "p implies q"	The conditional: false only when p is true and q false	$x > 2 \to x > 1$	Volume 3, Chapter 21
$p \leftrightarrow q$	"p if and only if q", "p iff q"	Same truth value	n even $\leftrightarrow$ n^2 even	Volume 3, Chapter 21
$\Rightarrow$	"implies"	Used between statements in an argument: "from this, it follows that". Many authors use $\Rightarrow$ and $\to$ interchangeably	$2x = 6 \Rightarrow x = 3$	Volume 3, Chapter 21
$\Leftrightarrow$	"if and only if", "is equivalent to"	Each side implies the other; an equation-solving step that can be reversed	$x + 1 = 3 \Leftrightarrow x = 2$	Volume 3, Chapter 21

Symbol	Read aloud	Meaning	Example	Ch.
$\forall$	"for all", "for every"	Universal quantifier	$\forall x \in \mathbb{R},\ x^2 \geq 0$	Volume 3, Chapter 21
$\exists$	"there exists", "for some"	Existential quantifier: at least one	$\exists x \in \mathbb{R},\ x^2 = 2$	Volume 3, Chapter 21
$\exists!$	"there exists a unique"	Exactly one	$\exists! x,\ 2x = 6$	Volume 3, Chapter 21
$\mathbf{T},\ \mathbf{F}$	"true", "false"	Truth values (also $1, 0$ or $\top, \bot$)		Volume 3, Chapter 21

C.5. Functions

Symbol	Read aloud	Meaning	Example	Ch.
$f(x)$	"f of x"	The output of function f at input x (not f times x)	$f(x) = x^2,\ f(3) = 9$	Volume 1, Chapter 11
$f : A \to B$	"f from A to B"	f takes inputs in A (domain) to outputs in B (codomain); a type signature	$\sqrt{\cdot} : [0, \infty) \to \mathbb{R}$	Volume 1, Chapter 11
$x \mapsto f(x)$	"x maps to f of x"	Names a function by its rule (like a lambda)	$x \mapsto 2x + 1$	Volume 1, Chapter 11
$f \circ g$	"f composed with g", "f after g", "f of g"	$(f \circ g)(x) = f(g(x))$: apply g first	$f(x) = x^2,\ g(x) = x + 1 \Rightarrow (f \circ g)(2) = 9$	Volume 1, Chapter 11
f^{-1}	"f inverse"	The function that undoes f. **Not** $1/f$	$f(x) = 2x \Rightarrow f^{-1}(x) = x/2$	Volume 1, Chapter 11
$\sin^{-1} x$, $\arcsin x$	"inverse sine of x", "arc sine of x"	The angle whose sine is x. But $\sin^2 x$ means $(\sin x)^2$	$\arcsin \frac{1}{2} = \frac{\pi}{6}$	Volume 1, Chapter 13
$\log_b x$, $\ln x$, $\log x$	"log base b of x", "natural log of x" ("ell-en x", "lin x" or "log x")	The exponent that gives x. $\ln$ is base e; bare $\log$ is base 10 in school, base e in research papers and in Python, often base 2 in CS	$\log_2 8 = 3$	Volume 1, Chapter 12
$\exp(x)$	"exp of x"	e^x, written so the exponent can be large	$\exp(0) = 1$	Volume 1, Chapter 12

Symbol	Read aloud	Meaning	Example	Ch.
Δx	"delta x"	A change (difference) in x	$\Delta x = x_2 - x_1$	Volume 1, Chapter 5
$F(x)\|_a^b$	"F of x evaluated from a to b"	$F(b) - F(a)$	$x^2\|_1^3 = 9 - 1 = 8$	Chapter 19
$O(g)$, $\Theta(g)$, $\Omega(g)$	"big O of g", "big Theta", "big Omega"	Grows at most / exactly / at least as fast as g, up to a constant factor	Binary search is $O(\log n)$	Volume 1, Chapter 14

C.6. Sequences and sums

Symbol	Read aloud	Meaning	Example	Ch.
a_n	"a sub n", "a n"	The n-th term of a sequence (code: `a[n]`)	$a_n = 2n \Rightarrow a_3 = 6$	Volume 1, Chapter 14
$(a_n)_{n\geq 1}$	"the sequence a n"	The whole sequence, not one term	$(1/n)_{n\geq 1}$	Volume 1, Chapter 14
$\sum_{i=1}^{n} a_i$	"the sum from i equals 1 to n of a sub i"	$a_1 + a_2 + \cdots + a_n$; a `for` loop accumulating with `+`	$\sum_{i=1}^{4} i = 10$	Volume 1, Chapter 14
$\prod_{i=1}^{n} a_i$	"the product from i equals 1 to n of a sub i"	$a_1 \cdot a_2 \cdots a_n$; the same loop accumulating with $\times$	$\prod_{i=1}^{4} i = 24$	Volume 1, Chapter 14
$\binom{n}{k}$	"n choose k"	Number of k-element subsets of an n-element set; also written $C(n,k)$, ${}^nC_k$, ${}_nC_k$, nCr	$\binom{5}{2} = 10$	Volume 3, Chapter 24
$P(n,r)$	"the number of permutations of n things r at a time"	Ordered selections: $n!/(n-r)!$; also ${}^nP_r$, nPr	$P(5,2) = 20$	Volume 3, Chapter 24

C.7. Calculus

Symbol	Read aloud	Meaning	Example	Ch.
$\lim_{x \to a} f(x)$	"the limit as x tends to a of f of x"	The value $f(x)$ approaches as x gets close to (but not equal to) a	$\lim_{x \to 0} \frac{\sin x}{x} = 1$	Chapter 16
$x \to a^+$, $x \to a^-$	"x tends to a from above / below"	One-sided limits	$\lim_{x \to 0^+} \frac{1}{x} = \infty$	Chapter 16
$\varepsilon,\ \delta$	"epsilon", "delta"	Small positive tolerances in the formal definition of a limit		Chapter 16
$f'(x)$	"f prime of x"	The derivative of f at x (Lagrange notation)	$f(x) = x^3 \Rightarrow f'(x) = 3x^2$	Chapter 17
$f''(x)$, $f^{(n)}(x)$	"f double prime of x", "the n-th derivative of f"	Second and n-th derivatives; brackets show it is not a power	$(x^3)'' = 6x$	Chapter 17
$\frac{dy}{dx}$	"d y d x", "the derivative of y with respect to x"	Leibniz notation: rate of change of y per unit x. Not a fraction, though it often behaves like one	$y = x^2 \Rightarrow \frac{dy}{dx} = 2x$	Chapter 17
$\frac{d}{dx}$	"d by d x"	The operation "differentiate with respect to x"	$\frac{d}{dx}\sin x = \cos x$	Chapter 17
$\dot{x}$	"x dot"	Newton notation: derivative with respect to time (physics, mechanics)	$\dot{x} = v$	Chapter 17
$\frac{\partial f}{\partial x}$, f_x, $\partial_x f$	"partial f by partial x", "del f del x", "f sub x"	Partial derivative: differentiate in x, holding the other variables fixed	$f = x^2 y \Rightarrow f_x = 2xy$	Chapter 20
∇f	"grad f", "del f", "nabla f"	The gradient: the vector of all partial derivatives	$\nabla(x^2 + y^2) = (2x, 2y)$	Chapter 20
$\int f(x)\,dx$	"the integral of f of x d x"	Indefinite integral: the family of antiderivatives	$\int 2x\,dx = x^2 + C$	Chapter 19

Symbol	Read aloud	Meaning	Example	Ch.
$\int_a^b f(x)\,dx$	"the integral from a to b of f of x d x"	Definite integral: signed area under f between a and b (a number). The $\int$ is an elongated S, for "sum"	$\int_0^1 x\,dx = \frac{1}{2}$	Chapter 19
dx	"d x"	Marks the variable of integration; the infinitesimal width of a strip		Chapter 19

C.8. Vectors and matrices

Symbol	Read aloud	Meaning	Example	Ch.
$\mathbf{v}$, $\vec{v}$, $\underline{v}$	"vector v", "v"	A vector. Print uses bold; handwriting uses an arrow or underline (the UK schools' squiggle)	$\mathbf{v} = (3, 4)$	Chapter 26
v_i	"v sub i"	The i-th component (code: `v[i]`, but mathematics usually counts from 1)	$v_2 = 4$	Chapter 26
$\|\mathbf{v}\|$	"the norm of v", "the length of v"	Length (Euclidean norm unless a subscript says otherwise)	$\|(3, 4)\| = 5$	Chapter 26
$\hat{\mathbf{v}}$	"v hat"	Unit vector in the direction of $\mathbf{v}$; $\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$ are the axis directions	$\hat{\mathbf{v}} = (0.6, 0.8)$	Chapter 26
$\mathbf{u} \cdot \mathbf{v}$	"u dot v"	Dot (scalar, inner) product; also $\langle \mathbf{u}, \mathbf{v} \rangle$	$(1, 2) \cdot (3, 4) = 11$	Chapter 26
$\mathbf{u} \times \mathbf{v}$	"u cross v"	Cross product (3D only): a vector perpendicular to both	$\hat{\mathbf{i}} \times \hat{\mathbf{j}} = \hat{\mathbf{k}}$	Chapter 26
$\perp$	"is perpendicular to", "is orthogonal to"	At right angles (dot product 0)	$(1, 0) \perp (0, 1)$	Chapter 26
A, a_{ij}	"matrix A", "a i j"	Matrices are capital letters; a_{ij} is the entry in row i, column j		Chapter 27
A^T, $A^\top$, A'	"A transpose"	Rows become columns: $(A^T)_{ij} = a_{ji}$	$\begin{pmatrix} 1 & 2 \end{pmatrix}^T = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$	Chapter 27
A^{-1}	"A inverse"	The matrix with $AA^{-1} = I$, when it exists		Chapter 27
I, I_n	"the identity", "I n"	Square matrix with 1s on the diagonal, 0s elsewhere	$I\mathbf{x} = \mathbf{x}$	Chapter 27

Symbol	Read aloud	Meaning	Example	Ch.
$\det A$, $\lvert A \rvert$	"the determinant of A", "det A"	Signed area/volume scale factor of A	$\det \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} = 6$	Chapter 27
$\operatorname{tr} A$	"the trace of A"	Sum of the diagonal entries	$\operatorname{tr} I_3 = 3$	Chapter 28
λ	"lambda"	An eigenvalue: $A\mathbf{v} = \lambda\mathbf{v}$		Chapter 28

C.9. Probability and statistics

Symbol	Read aloud	Meaning	Example	Ch.
Ω	"omega", "the sample space"	The set of all possible outcomes	Coin: $\Omega = \{H, T\}$	Volume 3, Chapter 29
$P(A)$, $\Pr(A)$	"the probability of A"	A number in $[0, 1]$	$P(\text{even die}) = \frac{1}{2}$	Volume 3, Chapter 29
$P(A \mid B)$	"the probability of A given B"	Probability of A once you know B happened	$P(\text{six} \mid \text{even}) = \frac{1}{3}$	Volume 3, Chapter 29
X, x	"capital X", "little x"	Capital: a random variable; lower case: a particular value it takes	$P(X = x)$	Volume 3, Chapter 29
$X \sim D$	"X is distributed as D", "X follows D"	Names the distribution of X	$X \sim N(0, 1)$	Volume 3, Chapter 30
$E[X]$, $\mathbb{E}[X]$	"the expected value of X", "the expectation of X"	Probability-weighted average	Fair die: $E[X] = 3.5$	Volume 3, Chapter 30
$\text{Var}(X)$	"the variance of X"	$E[(X - \mu)^2]$: average squared distance from the mean	Fair coin (0/1): 0.25	Volume 3, Chapter 30
μ	"mu"	Population (true) mean		Volume 3, Chapter 30

Symbol	Read aloud	Meaning	Example	Ch.
σ, σ^2	"sigma", "sigma squared"	Population standard deviation, variance		Volume 3, Chapter 30
$\text{Cov}(X, Y)$, ρ	"the covariance of X and Y", "rho"	How two variables vary together; ρ is the correlation		Volume 3, Chapter 30
$\bar{x}$	"x bar"	Sample mean	$\overline{\{2, 4, 9\}} = 5$	Volume 3, Chapter 31
s, s^2	"s", "s squared"	Sample standard deviation, sample variance		Volume 3, Chapter 31
$\hat{p}, \hat{y}, \hat{\theta}$	"p hat", "y hat", "theta hat"	An estimate or prediction of the un-hatted quantity	45 of 100 say yes: $\hat{p} = 0.45$	Volume 3, Chapter 31
H_0, H_1	"H nought", "H one"	Null and alternative hypotheses	$H_0 : \mu = 100$	Volume 3, Chapter 31
α	"alpha"	Significance level of a test	$\alpha = 0.05$	Volume 3, Chapter 31
z^*, t^*	"z star", "t star"	Critical values for confidence intervals	$z^* = 1.96$ for 95%	Volume 3, Chapter 31

C.10. Number theory and complex numbers

Symbol	Read aloud	Meaning	Example	Ch.
$a \mid b$	"a divides b"	b is a multiple of a (a statement, not a number)	$3 \mid 12, 5 \nmid 12$	Volume 1, Chapter 1
$a \bmod m$	"a mod m"	The remainder on dividing a by m (code: `a % m`)	$17 \bmod 5 = 2$	Volume 3, Chapter 23
$a \equiv b \pmod m$	"a is congruent to b mod m"	$m \mid (a-b)$: same remainder	$38 \equiv 2 \pmod{12}$	Volume 3, Chapter 23
$\gcd(a,b)$, $\operatorname{lcm}(a,b)$	"g c d of a and b", "l c m"	Greatest common divisor (UK: HCF), least common multiple	$\gcd(12,18) = 6$	Volume 1, Chapter 1
$\varphi(n)$	"phi of n", "Euler's totient of n"	How many of $1, \dots, n$ are coprime to n	$\varphi(10) = 4$	Volume 3, Chapter 23
i	"i"	$\sqrt{-1}$; engineers write j, as does Python (`1j`)	$i^2 = -1$	Volume 1, Chapter 15
$\bar{z}$, z^*	"z bar", "the conjugate of z"	Flip the sign of the imaginary part	$\overline{3+4i} = 3-4i$	Volume 1, Chapter 15
$\lvert z \rvert$, $\arg z$	"mod z", "arg z"	Modulus (length) and argument (angle) of z	$\lvert 3+4i \rvert = 5$	Volume 1, Chapter 15

Symbol	Read aloud	Meaning	Example	Ch.
$\operatorname{Re} z$, $\operatorname{Im} z$	"real part of z", "imaginary part of z"	a and b in $z = a + bi$	$\operatorname{Im}(3 + 4i) = 4$	Volume 1, Chapter 15

C.11. Optimisation and machine learning

Symbol	Read aloud	Meaning	Example	Ch.
$\min_x f(x)$, $\max_x f(x)$	"the minimum over x of f of x"	The smallest (largest) *value* of f	$\min_x (x-3)^2 + 1 = 1$	Chapter 18
$\operatorname{argmin}_x f(x)$, $\operatorname{argmax}_x f(x)$	"the arg min over x of f of x"	The *input* x that achieves the minimum (maximum)	$\operatorname{argmin}_x (x-3)^2 + 1 = 3$	Volume 3, Chapter 33
θ^*, x^*	"theta star"	The optimal value of a parameter		Volume 3, Chapter 33
η	"eta"	Learning rate (step size) in gradient descent	$\eta = 0.01$	Chapter 20
$\leftarrow$	"becomes", "gets"	Assignment in pseudocode (code: =)	$x \leftarrow x - \eta f'(x)$	Chapter 32

C.12. The Greek alphabet

Greek letters are used because the Latin ones run out, and certain letters have settled into conventional jobs. Knowing the conventions lets you guess what a symbol means before you're told.

Lower	Upper	Name	Common uses
α	A	alpha	Angles; significance level in statistics; coefficients; sometimes learning rate
β	B	beta	Angles; regression coefficients; Beta distribution; momentum in optimisation
γ	Γ	gamma	Angles; Euler-Mascheroni constant $\gamma \approx 0.5772$; discount factor in reinforcement learning. Γ: gamma function ($\Gamma(n) = (n-1)!$)
δ	Δ	delta	δ: small quantity (ε-δ limits), Kronecker delta. Δ: change in (Δx), discriminant
ε, ϵ	E	epsilon	Small positive quantity; error or noise term; machine epsilon
ζ	Z	zeta	Riemann zeta function $\zeta(s) = \sum 1/n^s$; damping ratio
η	H	eta	Learning rate; efficiency
θ, ϑ	Θ	theta	Angles; model parameters. Θ: asymptotically tight bound in algorithm analysis
ι	I	iota	Rare in mathematics; integer ranges in APL and Go
κ	K	kappa	Curvature; condition number of a matrix; Cohen's kappa
λ	Λ	lambda	Eigenvalues; rate of a Poisson or exponential distribution; Lagrange multiplier; wavelength; anonymous functions (lambda calculus). Λ: diagonal matrix of eigenvalues
μ	M	mu	Mean; micro- (10^{-6}); coefficient of friction
ν	N	nu	Degrees of freedom; frequency
ξ	Ξ	xi	Auxiliary or random variables; the point "somewhere in between" in error formulas
o	O	omicron	Almost never used (indistinguishable from Latin o)
π	Π	pi	$3.14159\ldots$; permutations; policy in reinforcement learning; prime-counting function. Π: product
ρ	P	rho	Correlation; density; radius in polar coordinates

Lower	Upper	Name	Common uses
σ	Σ	sigma	Standard deviation; sigmoid function; singular values; permutations. Σ: summation; covariance matrix; diagonal matrix of singular values in the SVD
τ	T	tau	Time constant; torque; $\tau = 2\pi$; Kendall's rank correlation
υ	Υ	upsilon	Rare
φ, ϕ	Φ	phi	Angles (especially in spherical coordinates); Euler's totient; golden ratio $\varphi \approx 1.618$; standard normal density. Φ: standard normal cumulative distribution function
χ	X	chi	Chi-squared distribution χ^2; characteristic (indicator) function
ψ	Ψ	psi	Wave functions in physics; auxiliary angles and functions
ω	Ω	omega	Angular frequency; complex cube root of unity; an outcome. Ω: sample space; asymptotic lower bound; ohms

Pronunciation: most English speakers say "pie" (π), "fie" or "fee" (φ), "kye" (χ), "zay-ta" or "zee-ta" (ζ), "mew" (μ), "new" (ν), "ksee", "zye" or "sigh" (ξ). US and UK speakers differ on "zee-ta"/"zay-ta" and "bay-ta"/"bee-ta"; either is fine.

C.13. Same symbol, different meanings

Notation is reused shamelessly. When a symbol seems to make no sense, check whether it has switched jobs.

Symbol	Meanings
$\lvert\cdot\rvert$ (vertical bars)	Absolute value $\lvert x\rvert$; modulus of a complex number $\lvert z\rvert$; size of a set $\lvert A\rvert$; determinant $\lvert A\rvert$. A single bar $\mid$ means "divides" ($3 \mid 12$), "such that" ($\{x \mid x > 0\}$) or "given" ($P(A \mid B)$)
Prime $'$	Derivative (f'); a related but different object (A', "A prime"); complement (A'); transpose in statistics (X'); feet and minutes ($5'$)
Bar $\bar{x}$	Mean ($\bar{x}$); complex conjugate ($\bar{z}$); complement ($\overline{A}$); repeating decimal ($0.\overline{3}$); logical NOT in circuit diagrams
Hat $\hat{x}$	Estimate ($\hat{\theta}$); prediction ($\hat{y}$); unit vector ($\hat{\mathbf{v}}$)
Superscript -1	Inverse function (f^{-1}); reciprocal ($x^{-1} = 1/x$); inverse matrix (A^{-1}); modular inverse ($3^{-1} \equiv 5 \pmod 7$)
Superscript 2 on a function	$\sin^2 x = (\sin x)^2$, but $f^2(x)$ sometimes means $f(f(x))$; read the author's conventions
Σ	Summation; covariance matrix; matrix of singular values
π	The constant; a permutation; a policy; the prime-counting function
e	Euler's number $2.71828\ldots$; an event; an edge in a graph; an identity element; scientific notation in code (`1e-9` means 10^{-9})
$\times$	Multiplication; cross product; Cartesian product; dimensions (a 3×4 matrix)
$\equiv$	Identity; congruence; logical equivalence
(a, b)	An open interval; a point in the plane; an ordered pair; $\gcd(a, b)$ in older number-theory texts

C.14. How to read a formula

A dense formula is code with one-letter variable names and no comments. Read it the way you would read unfamiliar code: find the overall shape, identify every name, then trace the evaluation from the inside out.

Example 47. Procedure — decoding a formula

Input: a formula you don't immediately understand.

Steps:

1. **Find the top-level shape.** What is being defined or claimed? Is it "name = expression", an equation to solve, an inequality, an update rule?
2. **Name every symbol.** For each letter, decide: constant, variable, index, function, or operator? Use this appendix and the surrounding text. Which symbols are *bound* (indices of a $\sum$, the variable of an $\int$, the x under an argmin) and which are *free* (inputs the result depends on)?
3. **Read from the inside out**, as an interpreter would evaluate it.
4. **Say it aloud** in words, then in plain English.
5. **Translate it into code.**
6. **Try a tiny example** small enough to do by hand.

Output: a sentence you could say to a colleague, and a few lines of code that compute it.

C.14.1. Decoding 1: sample variance

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Shape. A definition: the name s^2 equals (a fraction) times (a sum).

Symbols. n is the number of data values (free). x_i is the i-th data value. $\bar{x}$ ("x bar") is their mean. i is a bound index: it exists only inside the $\sum$, which runs it from 1 to n. s^2 is a single name, "s squared": it is the square of the standard deviation s.

Inside out. $x_i - \bar{x}$ is how far value i is from the mean (its *deviation*). Squaring makes it non-negative and punishes large deviations more. The $\sum$ adds the squared deviations for every data value. Finally multiply by $\frac{1}{n-1}$: an average, except dividing by $n-1$ rather than n (the reason is in Volume 3, Chapter 31).

Aloud. "s squared equals one over n minus one, times the sum from i equals 1 to n of x sub i minus x bar, all squared."

Plain English. "The sample variance is (almost) the average squared distance of the data from its mean."

Code and tiny example. For the data 2, 4, 6: the mean is 4, the deviations are $-2, 0, 2$, their squares are $4, 0, 4$, the sum is 8, and $8/(3-1) = 4$.

PYTHON

```
xs = [2, 4, 6]
n = len(xs)
xbar = sum(xs) / n
s2 = sum((x - xbar) ** 2 for x in xs) / (n - 1)
print(s2)
```

```
4.0
```

C.14.2. Decoding 2: fitting a model by least squares

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\operatorname{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\theta}^T \mathbf{x}_i \right)^2$$

Shape. A definition: $\hat{\boldsymbol{\theta}}$ is the result of an argmin, that is, the input that makes some expression smallest.

Symbols. The hat says $\hat{\boldsymbol{\theta}}$ is an *estimate*. Bold says $\boldsymbol{\theta}$ and $\mathbf{x}_i$ are vectors. $\boldsymbol{\theta} \in \mathbb{R}^d$ says $\boldsymbol{\theta}$ is a list of d real numbers (the model's parameters) and that the search ranges over all such lists; $\boldsymbol{\theta}$ is bound by the argmin. $\mathbf{x}_i$ is the i-th input (a feature vector) and y_i the matching observed output. $\boldsymbol{\theta}^T \mathbf{x}_i$ is a transpose times a vector, which is just the dot product $\boldsymbol{\theta} \cdot \mathbf{x}_i$: the model's prediction for example i.

Inside out. $y_i - \boldsymbol{\theta}^T \mathbf{x}_i$ is the *residual*, the truth minus the prediction. Square it, sum over all n examples, divide by n: the mean squared error of the model with parameters $\boldsymbol{\theta}$. The argmin then asks for the parameter vector that makes this error smallest.

Aloud. "Theta hat equals the arg min, over theta in R d, of one over n times the sum from i equals 1 to n of y sub i minus theta transpose x sub i, squared."

Plain English. "Our fitted parameters are whichever parameters make the average squared prediction error on the data as small as possible." This is linear regression (Volume 3, Chapter 31); the argmin is found either exactly by linear algebra (Volume 3, Section 33.2) or by gradient descent (Chapter 20, Volume 3, Chapter 33).

Code and tiny example. With $d = 1$ and two data points $(x, y) = (1, 2)$ and $(2, 4)$, the loss is $\frac{1}{2}\left((2 - \theta)^2 + (4 - 2\theta)^2\right)$, which is zero at $\theta = 2$, so $\hat{\theta} = 2$. A crude argmin by search confirms it:

PYTHON

```
X = [[1.0], [2.0]]
Y = [2.0, 4.0]

def loss(theta):
    n = len(X)
    return sum((y - sum(t * xj for t, xj in zip(theta, x))) ** 2
               for x, y in zip(X, Y)) / n

candidates = [[k / 100] for k in range(0, 501)]
theta_hat = min(candidates, key=loss)   # argmin by brute force
print(theta_hat, loss(theta_hat))
```

```
[2.0] 0.0
```

Notice that Python's `min(…, key=loss)` returns the *argument*, not the minimum value: it is an argmin. `loss(theta_hat)` is the min.

Appendix D: The US Mathematics Curriculum

If you were educated in the UK, the American system can be baffling: maths is called "math", it is taught as a sequence of separately named year-long courses rather than as one two-year syllabus, and a fifteen-year-old can be "doing Algebra II" or "in AP Calc" depending on which track they are on. This appendix is a translation guide. It explains the structure, compares it with GCSE and A-level, and maps each US course to the chapters of this book that cover it.



Read this as a cross-reference, not a rulebook

The United States has no national curriculum. Each of the 50 states sets its own standards, each of roughly 13,000 school districts chooses its own courses, textbooks and pacing, and individual schools offer different tracks. Everything below describes the *typical* pattern. Course names, grade levels and content vary from place to place; when it matters (for example, when choosing a course for next year), check your school's own course catalogue.

D.1. Grades, ages, and school levels

US school years are called **grades**. A child usually starts **kindergarten** (K) in the year they turn 5 or 6 (the cut-off date varies by state) and finishes **12th grade** at 17 or 18. Because the English Reception year comes before Year 1, US grade n corresponds roughly to English Year $n + 1$.

US grade	Typical age during the year	School level	Rough English equivalent
Pre-K	4-5	Preschool	Reception
K	5-6	Elementary school	Year 1
1-5	6-11	Elementary school	Years 2-6 (KS1-KS2)
6	11-12	Middle school (sometimes elementary)	Year 7
7-8	12-14	Middle school or junior high	Years 8-9 (KS3)
9 ("freshman")	14-15	High school	Year 10
10 ("sophomore")	15-16	High school	Year 11 (GCSE exams)
11 ("junior")	16-17	High school	Year 12 (Lower Sixth)
12 ("senior")	17-18	High school	Year 13 (Upper Sixth; A-level exams)

So a US 15-year-old is usually in 9th or 10th grade, the equivalent of the two GCSE years.

D.2. How US high-school mathematics is organised

The biggest difference from the UK is structural.

Courses, not syllabuses. In England, mathematics from 14 to 16 is one two-year GCSE course ending in national exams, and from 16 to 18 one two-year A-level course ending in national exams. In the US, each school year is a separately named **course** (Algebra I, Geometry, Algebra II, …), each lasting one year, each graded by the teacher, and each recorded on the student's **transcript**.

Continuous assessment. The course grade (A to F) is built from homework, quizzes, unit tests and a final exam set by the school. These grades feed into a **grade point average** (GPA), which universities weigh heavily. There is no national exit exam in mathematics, although many states set **end-of-course** (EOC) tests, especially for Algebra I.

Breadth until 18. US students keep studying six or seven subjects until they graduate. There is no equivalent of choosing three or four A-levels at 16. High schools typically require three or four years of mathematics to graduate (commonly including Algebra I and Geometry, often Algebra II), and selective universities expect four years, ideally reaching Precalculus or Calculus.

Tracks. Students in the same grade can be in very different courses. Placement into an **accelerated** track usually happens around grades 6-7 and determines how far a student gets by 12th grade.

D.3. The typical course sequence

Grade	Standard track	Accelerated track	Highly accelerated
6	Math 6	Math 6 (advanced)	Accelerated Math 6
7	Math 7	Accelerated Math 7 / Pre-Algebra	Pre-Algebra or Algebra I
8	Math 8 / Pre-Algebra	Algebra I	Algebra I or Geometry
9	Algebra I	Geometry (Honors)	Geometry or Algebra II (Honors)
10	Geometry	Algebra II (Honors)	Algebra II or Precalculus (Honors)
11	Algebra II	Precalculus (Honors or AP)	AP Calculus BC
12	Precalculus, Statistics, or another elective	AP Calculus AB or BC; AP Statistics	Multivariable Calculus, Linear Algebra (often via dual enrollment); AP Statistics

The "standard" route is the classic sequence

$$\textbf{Pre-Algebra} \to \textbf{Algebra I} \to \textbf{Geometry} \to \textbf{Algebra II} \to \textbf{Precalculus} \to \textbf{Calculus}$$

and the tracks differ mainly in *when* a student starts it. Taking Algebra I in 8th grade (rather than 9th) is the usual gateway to reaching calculus in high school.

Integrated pathways. A minority of states and districts teach **Integrated Math I, II, III** (sometimes called Math 1-3 or Secondary Math I-III) instead of Algebra I / Geometry / Algebra II. Each year mixes algebra, geometry, functions and statistics, which is closer to the UK approach. The total content over three years is roughly the same.



Why does Geometry get a whole year?

The US "layer-cake" sequence, with a year of Euclidean geometry sandwiched between two years of algebra, dates back to curriculum recommendations made in the 1890s, when Euclid was still the model of rigorous thinking. It survives today, and it has an upside: the US Geometry course is where most American students first write formal proofs, often in a distinctive **two-column** format (statement on the left, reason on the right). A typical UK student meets far less formal proof before A-level.

D.4. Common Core, in brief

The **Common Core State Standards for Mathematics** (2010) are a set of grade-by-grade expectations that most states adopted in the early 2010s. A few states never adopted them (Texas and Virginia, for example, have their own standards), and several have since revised or renamed them, but most state standards still closely resemble Common Core.

Key points for a parent:

- They are **standards** (what a student should be able to do by the end of each grade), not a curriculum or a textbook. Districts choose their own materials.
- Grades K-8 are specified year by year. High school is specified by topic area (Number and Quantity, Algebra, Functions, Modeling, Geometry, Statistics and Probability), so the same standards can be delivered as the traditional sequence or as an integrated one.
- Standards marked (+) are extra topics for students heading towards calculus and STEM subjects, such as complex-number arithmetic, vectors and matrices.
- Eight **Standards for Mathematical Practice** run through every grade: make sense of problems and persevere; reason abstractly and quantitatively; construct viable arguments; model with mathematics; use tools strategically; attend to precision; look for and use structure; look for regularity in repeated reasoning.
- What parents most often notice is the emphasis on multiple strategies and representations (number lines, area models for multiplication, "explain your reasoning"). That emphasis is close in spirit to this book's insistence on connecting words, symbols, pictures and computation.

D.5. Honors, AP, and IB

Honors. An Honors course covers the same course name at greater depth and pace, with harder problems. Many schools give Honors and AP grades extra weight in the GPA.

Advanced Placement (AP). AP courses follow a national syllabus set by the College Board and end in a standardised exam in May, scored 1 to 5. Many universities give credit or advanced placement for a 4 or 5 (some for a 3). AP is the closest US analogue to an A-level: a national, externally assessed exam in one subject. The AP mathematics courses are:

Course	Content	Chapters
AP Precalculus	Polynomial and rational functions; exponential and logarithmic functions; trigonometric and polar functions. (Parametric functions, vectors and matrices appear in the course but not on the exam.)	Volume 1, Chapter 11 , Volume 1, Chapter 7 , Volume 1, Chapter 12 , Volume 1, Chapter 13
AP Calculus AB	Limits and continuity; derivatives and their rules; implicit differentiation; related rates; L'Hôpital's rule; optimisation and curve sketching; Riemann sums; definite and indefinite integrals; the Fundamental Theorem; integration by substitution; slope fields and separable differential equations; area and volume. Roughly a first-semester university calculus course.	Chapter 16, Chapter 17, Chapter 18, Chapter 19
AP Calculus BC	Everything in AB, plus integration by parts and partial fractions; improper integrals; Euler's method; logistic growth; parametric, polar and vector-valued functions; infinite series (convergence tests, Taylor and Maclaurin series). Roughly two semesters of university calculus. The BC exam also reports an AB sub-score.	As AB, plus Volume 1, Chapter 14 , Chapter 32
AP Statistics	Exploring and describing data; sampling and experimental design; probability; random variables; binomial and geometric distributions; the normal distribution; sampling distributions; confidence intervals and significance tests (means, proportions, chi-square, regression slope). Roughly an introductory, non-calculus university statistics course.	Volume 3, Chapter 29 , Volume 3, Chapter 30 , Volume 3, Chapter 31 , Volume 3, Chapter 24

Related AP courses that use mathematics heavily: **AP Computer Science A** (Java programming), **AP Computer Science Principles**, **AP Physics 1** (algebra-based mechanics) and **AP Physics C: Mechanics** (calculus-based mechanics).

International Baccalaureate (IB). Some US high schools offer the two-year IB Diploma in grades 11-12. Its mathematics comes in two flavours, **Analysis and Approaches** (more pure, calculus-heavy) and **Applications and Interpretation** (more statistics and modelling), each at Standard Level (SL) or Higher Level (HL). IB Analysis and Approaches HL is roughly comparable to A-level Mathematics plus some Further Mathematics.

Dual enrollment. High-school students can take courses at a local community college or university, earning credit towards both their high-school diploma and a university degree. This is how strong students continue after AP Calculus BC: **Multivariable Calculus**, **Linear Algebra**, **Differential Equations** and **Discrete Mathematics** are the usual next courses.

D.6. SAT and ACT mathematics

Most US universities consider (and some require) a standardised admissions test, the **SAT** or the **ACT**. Both include mathematics, and both test the content of Algebra I, Geometry and parts of Algebra II, with no calculus. Their formats change from time to time, so check the current specifications, but at the time of writing:

- **SAT Math** (College Board): a computer-adaptive section in two modules, 44 questions in 70 minutes, scored 200-800. A calculator (including a built-in graphing calculator) is allowed throughout. Four content areas: Algebra (linear equations, inequalities and systems); Advanced Math (quadratics, polynomials, exponentials, function notation); Problem-Solving and Data Analysis (ratios, percentages, units, statistics, probability); Geometry and Trigonometry. Relevant chapters: Volume 1, Chapter 2 to Volume 1, Chapter 10 , the start of Volume 1, Chapter 13 , and the descriptive parts of Volume 3, Chapter 31 .
- **ACT Math**: 45 multiple-choice questions in 50 minutes, scored 1-36, calculator allowed. Slightly broader than the SAT, with occasional logarithms, matrices, complex numbers and trigonometry from Algebra II and Precalculus.
- The **PSAT/NMSQT**, taken in 10th or 11th grade, is a shorter version of the SAT and the qualifying test for National Merit Scholarships.

D.7. The UK and US systems side by side

UK stage	Age	Closest US counterpart	Chapters
Key Stage 3 (Years 7-9)	11-14	Math 6, 7, 8 and Pre-Algebra; Algebra I for accelerated students	Volume 1, Chapter 1 to Volume 1, Chapter 5
GCSE Mathematics, Foundation tier	14-16	Pre-Algebra and Algebra I, with basic geometry and statistics	Volume 1, Chapter 1 to Volume 1, Chapter 6 , Volume 1, Chapter 9
GCSE Mathematics, Higher tier	14-16	Algebra I plus Geometry plus parts of Algebra II: quadratics, simultaneous equations, functions, right-triangle trigonometry, sine and cosine rules, circle theorems, basic vectors, probability. No calculus beyond informal gradients and areas under graphs	Volume 1, Chapter 4 to Volume 1, Chapter 10 , start of Volume 1, Chapter 13 , Volume 3, Chapter 29
Level 2 Further Maths / Additional Maths	15-16	Honors Algebra II and early Precalculus	Volume 1, Chapter 7 , Volume 1, Chapter 11 , Volume 1, Chapter 13
A-level Mathematics (pure)	16-18	Algebra II, Precalculus and most of AP Calculus BC: proof, functions, logs and exponentials, trigonometric identities, sequences and series, differentiation and integration (including by parts and substitution), parametric equations, simple differential equations, numerical methods (Newton-Raphson, trapezium rule)	Volume 1, Chapter 11 to Chapter 19, Volume 3, Chapter 22 , Chapter 32
A-level Mathematics (statistics)	16-18	Much of AP Statistics: sampling, binomial and normal distributions, hypothesis testing, correlation and regression	Volume 3, Chapter 29 to Volume 3, Chapter 31
A-level Mathematics (mechanics)	16-18	Taught in *physics* in the US: AP Physics 1, or AP Physics C: Mechanics for the calculus-based version	Chapter 26, Chapter 17, Chapter 19

UK stage	Age	Closest US counterpart	Chapters
A-level Further Mathematics	16-18	Beyond the AP courses: complex numbers (including De Moivre), matrices, proof by induction, polar coordinates, hyperbolic functions, series, further calculus and differential equations. Comparable to first-year university Linear Algebra and Differential Equations, reached in the US by dual enrollment	Volume 1, Chapter 15 , Chapter 27, Chapter 28, Volume 3, Chapter 22 , Volume 1, Chapter 14

Some broad differences are worth keeping in mind.

- **Specialisation.** An A-level student studies mathematics for a large fraction of the school week from 16. A US student spends one period a day on it. In compensation, the US student usually also studies English, history, a science and a language to 18.
- **Calculus timing.** An A-level student meets calculus at 16-17. A US student on the standard track may not meet it until university; an accelerated student meets it at 16-17, much like the A-level student.
- **Proof and geometry.** US students meet formal two-column proof earlier (in Geometry, typically at 14-15). UK students meet more algebraic manipulation earlier.
- **Statistics.** Both systems now include it, but in the US it is often a separate course (AP Statistics) rather than a strand woven through every year.



What happened to "Maths with Mechanics"?

In the 1980s and early 1990s many English A-level students took **Mathematics with Mechanics**, or **Pure and Applied Mathematics**: pure mathematics (algebra, trigonometry, calculus) alongside applied mathematics, which in practice meant Newtonian mechanics (kinematics, forces, Newton's laws, projectiles, moments, energy). The US has no single course like it. The pure half corresponds to Algebra II, Precalculus and AP Calculus; the mechanics half lives in the physics department, where **AP Physics C: Mechanics** is the closest match because it uses calculus.

D.8. Terminology: UK and US words for the same things

UK	US	Notes
maths	math	
Year 11	10th grade	Subtract one from the English year number
BIDMAS / BODMAS	PEMDAS	US mnemonic "Please Excuse My Dear Aunt Sally": Parentheses, Exponents, Multiplication/Division, Addition/Subtraction
brackets	parentheses	US "brackets" usually means square brackets [ ]
highest common factor (HCF)	greatest common factor (GCF) or greatest common divisor (GCD)	
index, indices, powers	exponent(s)	
surd	radical	
standard form (3.2×10^5)	scientific notation	US "standard form" means a line written as $Ax + By = C$, or a number written out in full
$y = mx + c$	$y = mx + b$	The y-intercept is called b in the US
gradient (of a line)	slope	
simultaneous equations	system of equations	
recurring decimal	repeating decimal	
modulus $\lvert x \rvert$	absolute value	"Modulus" is still used for complex numbers
trapezium	trapezoid	Same shape: a quadrilateral with (at least) one pair of parallel sides. Confusingly, in older usage each country's *other* word meant a quadrilateral with no parallel sides
turning point	vertex (of a parabola), extremum	

UK	US	Notes
stationary point	critical point	US "critical point" also includes points where the derivative is undefined
sine rule, cosine rule	law of sines, law of cosines	
Pythagoras' theorem	the Pythagorean theorem	
Newton-Raphson	Newton’s method	
trapezium rule	trapezoidal rule	
sum to infinity	sum of an infinite (geometric) series	
differentiation from first principles	using the limit definition of the derivative	
nought	zero	
marks	points	

D.9. Course-by-course map to this book

Course	Typical grade (standard / accelerated)	Key topics	Chapters
Elementary math	K-5	Place value, the four operations, multiplication tables, fractions (from grade 3), decimals, measurement, simple geometry, simple data displays	Volume 1, Chapter 1 , Volume 1, Chapter 2 , start of Volume 1, Chapter 9
Math 6 and Math 7	6-7	Ratios, rates and percentages; negative numbers; expressions; one-variable equations and inequalities; area, surface area and volume; introductory statistics and probability	Volume 1, Chapter 1 , Volume 1, Chapter 2 , Volume 1, Chapter 4 , start of Volume 1, Chapter 5 , Volume 1, Chapter 9
Math 8 / Pre-Algebra	8 / 7	Exponents and scientific notation; square roots and irrational numbers; linear equations and functions; introductory systems; the Pythagorean theorem; transformations	Volume 1, Chapter 3 , Volume 1, Chapter 5 , Volume 1, Chapter 6 , Volume 1, Chapter 11 , Volume 1, Chapter 9
Algebra I	9 / 8	Linear equations and inequalities; slope and lines; functions; systems; exponent rules; exponential functions (introduction); adding, multiplying and factoring polynomials; quadratics; data and scatter plots	Volume 1, Chapter 4 , Volume 1, Chapter 5 , Volume 1, Chapter 6 , start of Volume 1, Chapter 7 , Volume 1, Chapter 8 , Volume 1, Chapter 11
Geometry	10 / 9	Congruence, similarity and transformations; two-column proof; right-triangle trigonometry; circles; coordinate geometry; area and volume	Volume 1, Chapter 9 , Volume 1, Chapter 10 , start of Volume 1, Chapter 13 , Volume 3, Chapter 22

Course	Typical grade (standard / accelerated)	Key topics	Chapters
Algebra II	11 / 10	Polynomial and rational functions; radicals; complex numbers; exponential and logarithmic functions; sequences and series; inverse functions and transformations; often the unit circle and trigonometric functions; probability and statistics	Volume 1, Chapter 7 , Volume 1, Chapter 11 , Volume 1, Chapter 12 , Volume 1, Chapter 15 , Volume 1, Chapter 14 , Volume 1, Chapter 13 , Volume 3, Chapter 29
Precalculus (including AP Precalculus)	12 / 11	Functions in depth; trigonometric functions, identities and equations; polar coordinates and complex numbers in polar form; vectors; sometimes matrices and conic sections; sequences, series and an introduction to limits	Volume 1, Chapter 11 , Volume 1, Chapter 13 , Volume 1, Chapter 15 , Volume 1, Chapter 10 , Volume 1, Chapter 14 , Chapter 26, Chapter 27, Chapter 16
AP Calculus AB	12 / 11	See the AP table above	Chapter 16, Chapter 17, Chapter 18, Chapter 19
AP Calculus BC	12 / 11	AB plus series, parametric and polar calculus, further integration	As AB, plus Volume 1, Chapter 14 , Chapter 32
AP Statistics (or non-AP Statistics)	11-12	Data, sampling and experiments, probability, distributions, inference	Volume 3, Chapter 24 , Volume 3, Chapter 29 , Volume 3, Chapter 30 , Volume 3, Chapter 31
Discrete Mathematics (occasional high-school elective; standard at university)	12 / university	Logic, sets, proof and induction, number theory, counting, graphs	Volume 3, Chapter 21 , Volume 3, Chapter 22 , Volume 3, Chapter 23 , Volume 3, Chapter 24 , Volume 3, Chapter 25
Multivariable Calculus (Calculus III)	post-BC / university	Partial derivatives, gradients, optimisation in several variables, multiple integrals, vector calculus	Chapter 20 (the book stops before multiple integrals and vector calculus)

Course	Typical grade (standard / accelerated)	Key topics	Chapters
Linear Algebra	post-BC / university	Vectors, matrices, systems, determinants, vector spaces, eigenvalues, orthogonality, the SVD	Volume 1, Chapter 6 , Chapter 26, Chapter 27, Chapter 28

D.10. Quick reference: what does "He's doing ..." mean?

"She's in Math 7"

Ratios, percentages, negative numbers, simple equations. UK Year 8. Chapters Volume 1, Chapter 2 , Volume 1, Chapter 4 .

"He's doing Pre-Algebra"

The bridge from arithmetic to symbols: exponents, roots, simplifying expressions, solving linear equations, the Pythagorean theorem. UK Years 8-9. Chapters Volume 1, Chapter 3 , Volume 1, Chapter 4 , Volume 1, Chapter 5 .

"She's taking Algebra I in 8th grade"

She is on an accelerated track, one year ahead of the standard sequence, and on course for calculus in high school. Content: lines, systems, functions, factoring, quadratics. Roughly the algebra of GCSE Higher. Chapters Volume 1, Chapter 5 to Volume 1, Chapter 8 , Volume 1, Chapter 11 .

"He's doing Geometry"

A full year of Euclidean geometry with formal proof, right-triangle trigonometry, circles and coordinate geometry. Roughly the geometry of GCSE Higher, plus far more proof. Chapters Volume 1, Chapter 9 , Volume 1, Chapter 10 , Volume 3, Chapter 22 .

"He's doing Algebra II"

The big pre-calculus algebra course: polynomials, rational functions, complex numbers, exponentials and logarithms, sequences and series, often trigonometric functions. In UK terms, the top end of GCSE Higher plus a good part of the pure algebra in the first year of A-level. A 15-year-old in Algebra II is a year or two ahead of the standard track. Chapters Volume 1, Chapter 7 , Volume 1, Chapter 11 , Volume 1, Chapter 12 , Volume 1, Chapter 15 , Volume 1, Chapter 14 .

"She's in Honors Algebra II"

The same territory, faster and with harder problems; sometimes includes topics normally saved for Precalculus.

"He's doing Integrated Math II"

A mixed course: typically quadratics, more functions, similarity and right-triangle trigonometry, probability. Roughly the second half of Algebra I combined with much of Geometry.

"She's doing Precalc"

Functions in depth, full trigonometry, polar form, vectors, an introduction to limits. Roughly A-level pure mathematics without the calculus. Chapters Volume 1, Chapter 11 , Volume 1, Chapter 13 , Volume 1, Chapter 14 , Chapter 16.

"He's in AP Calc AB" / "She's taking BC"

University-level single-variable calculus, ending in a national exam in May. AB is roughly the calculus of A-level Mathematics; BC adds series and polar and parametric calculus, taking it a little beyond. Chapters Chapter 16 to Chapter 19.

"She got a 5 in BC"

The top score on the AP Calculus BC exam. Many universities will grant credit for one or two semesters of calculus.

"He's doing AP Stats"

Introductory statistics: data, sampling, probability, the normal distribution, confidence intervals and hypothesis tests. Roughly the statistics of A-level Mathematics. Chapters Volume 3, Chapter 29 to Volume 3, Chapter 31 .

"She's taking Multivariable at the community college"

She has finished AP Calculus BC early and is taking university-level calculus through dual enrollment. Chapter Chapter 20, and beyond this book.

"He got a 740 on the SAT Math"

A strong score (out of 800), reflecting secure command of Algebra I, Geometry and Algebra II material. It says nothing about calculus.

> ### US curriculum
>
> Every major topic in this book carries a short **US curriculum** sidebar like this one, naming the course where the topic typically appears. Those sidebars and the tables in this appendix are meant to work together: if your teenager says "we're doing logarithms in Algebra II", the sidebar in Volume 1, Chapter 12 tells you where you are, and this appendix tells you what comes next.

D.11. University mathematics for computer science

US computer-science degrees almost all require a common core of mathematics, usually taken in the first two years. (UK computer-science degrees cover much the same ground in first-year modules with names like "Mathematics for Computer Science" and "Discrete Mathematics".)

Course	Typical content	Chapters
Calculus I and II	Limits, derivatives, integrals, techniques of integration, sequences and series (AP Calculus BC with a 4 or 5 often exempts students from both)	Chapter 16 to Chapter 19, Volume 1, Chapter 14
Discrete Mathematics (one or two semesters)	Logic, sets, functions and relations, proof techniques and induction, recurrences, counting, number theory, graphs and trees. The most important mathematics course for most programmers	Volume 3, Chapter 21 , Volume 3, Chapter 22 , Volume 3, Chapter 23 , Volume 3, Chapter 24 , Volume 3, Chapter 25 , Volume 1, Chapter 14
Linear Algebra	Systems of equations, matrices, vector spaces, determinants, eigenvalues, orthogonality, least squares, the SVD. Essential for graphics, machine learning and scientific computing	Volume 1, Chapter 6 , Chapter 26, Chapter 27, Chapter 28
Probability and Statistics	Probability, random variables, distributions, expectation, the central limit theorem, estimation and testing; often calculus-based	Volume 3, Chapter 29 , Volume 3, Chapter 30 , Volume 3, Chapter 31
Multivariable Calculus (often optional for CS, required for machine learning, graphics and robotics specialisms)	Partial derivatives, gradients, optimisation, multiple integrals	Chapter 20, Volume 3, Chapter 33
Numerical Methods / Scientific Computing (elective)	Floating point, root finding, numerical integration and differential equations, numerical linear algebra	Chapter 32

Algorithms and theory-of-computation courses then draw on all of this, especially discrete mathematics, asymptotic notation and probability. Machine-learning courses lean hardest on linear algebra, multivariable calculus, probability and optimisation, which is exactly the combination brought together in Volume 3, Chapter 33 .

Appendix E: Interactive Lab Inventory

Every Interactive Lab opportunity marked in Volume 2, collected in one place. Each entry is a hands-on experiment that makes the idea in its chapter tangible. Most now have a working version on the companion website (https://geekychris.github.io/tools/): follow the entry's _Try it_ link to open the tool with that example already set up._

There are 32 labs across 9 chapters.

E.1. Chapter 16

Secant to tangent *(in: The problem: speed at an instant)*

Show the graph of $d(t) = 21t^2$ with a fixed point at $t = 1.5$ and a draggable second point. Display the secant line, its slope, and h. As the user drags the second point in from either side, a table records the slopes, and the secant visibly rotates into the tangent. Let the user change $d(t)$ to $\sqrt{t}$, $\sin t$, or $|t - 1.5|$ (where the left and right secants disagree).

Try it: Try $|t - 1.5|$: no tangent · Open the secant-to-tangent lab · Slide h: secant to tangent

Bisection stepper *(in: The intermediate value theorem and bisection)*

Enter a function and an interval. Each click on "step" shades the current interval, marks the midpoint with the sign of $f(m)$, and keeps the half containing the sign change. A side panel plots the interval width on a logarithmic scale (a straight line: halving each step). Include a mode with $1/x$ to watch bisection "converge" to an asymptote.

Try it: Bisection on 1/x · Open the bisection stepper

The ε–δ game *(in: Optional depth: the formal ε–δ definition)*

The app plays challenger: it shows a graph, a point a, a claimed limit L, and a horizontal band of half-height ε. The user drags a vertical strip of half-width δ; the portion of the graph inside the strip turns green if it fits within the band and red where it escapes. Each round, ε shrinks. For a function with a jump, the user discovers that no δ works once ε is smaller than half the jump — a visual proof that the limit does not exist.

Try it: Try to win at a jump · Play the ε–δ game

Floating-point limits *(in: Limits meet the computer)*

A slider for h from 10^{-1} to 10^{-20} (log scale) computes the difference quotient of a chosen function in double precision and in exact rational arithmetic side by side, plotting both errors on log–log axes. Show the bits of $x + h$ and x as h shrinks, highlighting the moment they become identical.

Try it: Floating-point limits, bit by bit

E.2. Chapter 17

Secant to tangent *(in: From secant to tangent)*

Drag a second point Q along a curve towards a fixed point P. A readout shows the secant slope to six decimal places as h shrinks, and a second panel plots secant slope against h. Try a smooth curve, then $|x|$ at $P = (0, 0)$, and watch the slopes from the left and the right refuse to agree.

Try it: Try |x| at the origin · Open the secant-to-tangent lab · Secants of |x| at 0

Finding the sweet spot ***(in: How computers compute derivatives)***

A slider sets $\log_{10} h$ from -16 to 0 and plots forward and central difference errors for a function you choose ($\sin$, e^x, x^3, $\sqrt{x}$ near 0). Find the best h for each. Then switch to 32-bit floats and watch the sweet spot move to much larger h.

Try it: Find the sweet spot · The sweet spot in float32 · Find the sweet spot

Build a computation graph ***(in: How computers compute derivatives)***

Drag nodes ($+$, $\times$, $\sin$, $\exp$, x^2) onto a canvas to build an expression graph. Press "forward" to watch values flow from inputs to output; press "backward" to watch local derivatives multiply their way back from the output, with each edge labelled by its local derivative. Compare the result with forward-mode dual numbers and with a finite difference.

Try it: Build a computation graph

E.3. Chapter 18

Box designer ***(in: Optimisation)***

Drag a slider for the cut size x on a 20 cm sheet and watch a 3-D box fold up, with its volume plotted live as a point on the $V(x)$ curve. A second slider changes the sheet to a rectangle ($a \times b$); see the optimum move, and compare it with the root of $V'(x) = 0$ found by the quadratic formula.

Try it: Open the box designer · Try a rectangular sheet · Box volume V(x) with sliders

Newton's method explorer ***(in: Newton's method)***

Adjust the starting value x_0 for Newton's method with a slider and observe which root it reaches. The curve is drawn with each tangent line and each new iterate animated, and a side panel shows a table of x_n, $f(x_n)$, $f'(x_n)$ and the correction. Preset functions: $x^2 - 2$ (calm), $x^3 - 2x + 2$ (find the start that cycles), $\arctan x$ (find the divergence threshold near 1.39), and $x^3 - x$ (colour the starting line by which root wins and zoom into the interleaved basins). Toggle "bisection" to race the two methods on the same function and plot their errors on a log scale. Extension: in the complex plane (Volume 1, Chapter 15), colour each starting point z_0 by the root of $z^3 - 1$ it reaches, and watch a fractal appear.

Try it: Colour x³ − x basins · Open the Newton explorer · The z³ − 1 fractal

E.4. Chapter 19

Rectangles converging ***(in: Riemann sums: rectangles under a curve)***

Pick a function and an interval. Drag a slider for n from 1 to 200 and switch between left, right and midpoint rules. Watch the rectangles and a live readout of the sum and its error against the exact area. Try a decreasing function and a function that rises and falls to test the "underestimate" slogan.

Try it: Open the rectangles lab · Left sums of a decreasing f · A function that rises and falls · Riemann rectangles, slide n

Painting the area ***(in: The Fundamental Theorem of Calculus)***

Show a curve $y = f(t)$ on top and, underneath, the graph of the accumulation function $A(x) = \int_a^x f(t)\,dt$. Drag the endpoint x and watch the shaded area and $A(x)$ update together. Show the tangent slope of A at x next to the height $f(x)$: they always match. Try a curve that dips below the axis and watch A decrease there.

Try it: Open the painting-the-area lab · f and its area function

Watching a substitution ***(in: Integration by parts: the product rule in reverse)***

Plot $2x\cos(x^2)$ on $[0, \sqrt{\pi}]$ beside $\cos u$ on $[0, \pi]$, with a slider that moves a thin strip in one plot and the corresponding strip in the other. The strips have different widths and heights but the same area: substitution is a stretching of the horizontal axis that preserves area.

Try it: Open the substitution lab

Racing the rules ***(in: Numerical integration)***

Choose a function from a menu (smooth, kinked, oscillating, spiking) and a number of strips. Show trapezoids, midpoint rectangles and Simpson parabolas overlaid on the curve, with a live log–log error plot growing as you increase n. Find a function where the trapezoid rule beats Simpson (hint: a periodic function over one full period, or one with a kink).

Try it: A kink slows Simpson · Race the rules

A rate that depends on itself ***(in: Differential equations: a first look)***

Draw a **slope field** for $dy/dt = g(t, y)$: a tiny line segment at each grid point showing the slope there. Click a starting point and watch Euler's method trace a path through the field with an adjustable Δt, alongside the exact solution where known. Try $dy/dt = 0.5y$, $dy/dt = -y$, and the logistic equation $dy/dt = y(1 - y)$.

Try it: Open the slope-field lab

E.5. Chapter 20

Explore a surface ***(in: Functions of several variables)***

Type a formula $f(x, y)$ and see three linked views: the rotatable surface, the contour plot and the heat map. Hover over a point in any view to highlight it in the other two. Change the contour spacing and watch steep regions show up as bunched lines.

Try it: Open the surface explorer

Follow the arrow ***(in: Directional derivatives and the direction of steepest ascent)***

Click a point on a contour map. The app draws ∇f there and a draggable unit vector $\mathbf{u}$. As you rotate $\mathbf{u}$, a live readout shows $D_{\mathbf{u}} f = |\nabla f| \cos\theta$ and a dot moves along the cosine wave. Confirm by hand that the maximum is along the gradient and the zeros are along the contour.

Try it: Open the Follow the arrow lab

Tune the learning rate ***(in: Gradient descent)***

Pick a surface (round bowl, elongated bowl, a banana-shaped Rosenbrock valley, a two-minimum landscape). Click to choose a starting point and drag a slider for η. The app animates the descent path on the contour plot and draws the loss curve beside it. Find the largest η that still converges on the elongated bowl, and check it against the prediction $2/\lambda_{\max}$. On the two-minimum landscape, colour each starting point by which minimum it reaches.

Try it: Colour the two basins · Open the learning-rate lab · Descend the Rosenbrock valley

E.6. Chapter 26

Drag the arrows ***(in: Adding and scaling vectors)***

Two draggable arrows $\mathbf{a}$ and $\mathbf{b}$ start at the origin. The app draws $\mathbf{a} + \mathbf{b}$ both tip-to-tail and as the parallelogram diagonal, and shows the component arithmetic updating live. A slider for c shows $c\mathbf{a}$ stretching, shrinking, and flipping as c passes through zero.

Try it: Drag the arrows

Combination sliders ***(in: Linear combinations and span)***

Fix two vectors $\mathbf{a}$ and $\mathbf{b}$. Two sliders control x and y, and the app draws $x\mathbf{a} + y\mathbf{b}$ with a faint trail. Sweep the sliders and watch the trail fill the plane. Then drag $\mathbf{b}$ onto the line of $\mathbf{a}$ and watch the reachable set collapse to a line. A "target" mode places a random point and asks the learner to find the coefficients.

Try it: Combination sliders, collapsing span

Resolve a force ***(in: Projections)***

Drag a vector $\mathbf{a}$ and a direction $\mathbf{b}$. The app shows the shadow of $\mathbf{a}$ on the line of $\mathbf{b}$, the perpendicular residual with its right-angle marker, and the numbers $\mathbf{a} \cdot \mathbf{b}$, k and θ. Rotate $\mathbf{a}$ past 90° and watch the projection flip to point backwards as the dot product goes negative. A "ramp" mode tilts a slope and resolves the box's weight live.

Try it: Resolve a force

Spin up the dimension ***(in: High-dimensional intuition, and where it fails)***

A slider sets the dimension n from 2 to 2000. The app draws 1000 random vector pairs and plots the histogram of their cosine similarities, with the curve narrowing as n grows. A second panel shows the distribution of distances from one random point to all others, so the learner can watch "nearest"

and "farthest" converge. A third loads real sentence embeddings for the learner's own sentences and compares their similarity scores against the random baseline.

Try it: Spin up the dimension

E.7. Chapter 27

Change a matrix, watch the grid *(in: Transformations of space)*

Four sliders (or draggable number fields) set the entries of a 2×2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$. The app shows the coordinate grid, the unit square, an asymmetric F and the two basis arrows, and animates them smoothly from the identity to the chosen transformation. The learner can also drag the tips of the green and red arrows directly, with the matrix entries updating to match (the columns **are** the arrow tips). Readouts show the determinant (with the unit square's area highlighted) and flag when it reaches zero and the plane collapses to a line. Preset buttons: rotate, scale, shear, reflect, project. A "compose" mode applies two matrices in sequence and lets the learner swap their order.

Try it: Change a matrix, watch the grid

Neurons as directions *(in: Neural network layers: matrix multiply plus a bend)*

A 2D dataset of two interleaved classes (for example two spirals). The learner picks a 2×2 matrix and bias for one layer and watches the data transform; then toggles ReLU on and off and sees how the bend folds the plane. A second layer follows. The app shows that without ReLU, two layers collapse to one matrix (it displays $W_2 W_1$), while with ReLU the classes can be pulled apart.

Try it: Neurons as directions: ReLU fold

E.8. Chapter 28

Experiment: Eigenvectors using Python *(in: Finding eigenvalues: the characteristic equation)*

Verify the worked example, then probe the failure modes.

```python
import numpy as np

A = np.array([[4.0, 1.0],
              [2.0, 3.0]])
vals, vecs = np.linalg.eig(A)
print(vals)
print(vecs)                    # eigenvectors are the COLUMNS, scaled to length 1

v = vecs[:, 0]
print(A @ v, vals[0] * v)      # A v equals lambda v
```

```
[5. 2.]
[[ 0.70710678 -0.4472136 ]
 [ 0.70710678  0.89442719]]
[3.53553391 3.53553391] [3.53553391 3.53553391]
```

The columns are $(1,1)/\sqrt{2}$ and $(-1,2)/\sqrt{5}$, which is $-(1,-2)/\sqrt{5}$: the same direction as our hand answer, normalised and flipped. Now try these and explain each result geometrically:

1. `np.linalg.eig(np.array([[0.0, -1.0], [1.0, 0.0]]))` (a rotation: expect complex output).
2. `np.linalg.eig(np.array([[1.0, 1.0], [0.0, 1.0]]))` (a shear: look at the two eigenvector columns; are they really different?).
3. A random symmetric matrix `M = R + R.T` for a random `R`: compute `vecs.T @ vecs` and see what you get (preview of Section 28.9).
4. Pick any 2×2 matrix and any starting vector, apply the matrix 30 times (normalising each time), and compare the result with the eigenvector of the largest eigenvalue.

Try it: Verify it step by step · Run the four experiments

Grow your own web ***(in: PageRank: the eigenvector that built Google)***

The learner draws pages as nodes and links as arrows. The app animates a crowd of random surfers hopping along links (with occasional teleports), sizes each node by the fraction of surfers on it, and shows the power-iteration vector converging alongside. Sliders: damping factor d from 0 to 1 (watch rankings flatten towards uniform as $d \to 0$, and convergence slow as $d \to 1$). Challenges: add a "link farm" of pages that all point at one page and see how much rank it gains; create a dead-end page and a closed loop and observe what goes wrong with $d = 1$.

Try it: Grow your own web · PageRank with a damping slider

Rotate the cloud ***(in: Principal component analysis)***

A scatter plot of 2D data with a draggable line through the centre. As the learner rotates the line, the app projects every point onto it and shows the variance of the projections as a live number and a dial. The learner hunts for the maximum, then clicks "reveal" to see PC1 drawn from the eigenvector of the covariance matrix, landing exactly on their best line. Further sliders change the correlation and the noise, so the learner sees the eigenvalue ratio (variance explained) track how elongated the cloud is.

Try it: Rotate the cloud

Compress it yourself ***(in: The singular value decomposition)***

The learner uploads a photo (or picks one). The app converts it to greyscale, computes its SVD and shows a slider for the rank k, redrawing the rank-k image live alongside a plot of the singular values with the cutoff marked, the storage cost, and the relative error. A second panel shows the individual rank-1 "layers" $\sigma_i \mathbf{u}_i \mathbf{v}_i^\mathsf{T}$ for $i = 1, 2, 3, \ldots$, so the learner sees the image being built from stripes. A third shows the unit circle turning into an ellipse for a user-chosen 2×2 matrix, step by step through V^T, Σ and U.

Try it: Unit square through V^T, Σ, U · Compress it yourself

E.9. Chapter 32

Float explorer *(in: Floating point: how real numbers are stored)*

Type any decimal number and see its 64 bits coloured by field, its exact stored value, the neighbouring doubles on either side, and its ulp. Drag a slider through magnitudes from 10^{-320} to 10^{308} and watch the spacing change, the subnormal region appear, and the value finally overflow to infinity.

Try it: Float explorer: bits and ulp · The smallest subnormal double

Newton's basins *(in: Root finding compared)*

Plot $f(x) = x^3 - 2x + 2$ and drag the starting point for Newton's method. From $x_0 = 0$ the iterates cycle $0 \to 1 \to 0 \to \dots$ forever. Colour each starting point by which root (if any) it reaches, then switch to the complex plane (Volume 1, Chapter 15) for $z^3 - 1 = 0$ and watch the famous fractal Newton basins appear.

Try it: Complex basins of $z^3 - 1$ · Cycle islands of $z^3 - 2z + 2$ · Open Newton's basins

Halving h *(in: Numerical differentiation: the V-shaped error)*

A slider sets $\log_{10} h$ from 0 to -16. Show the secant line on the graph of f, the computed slope, and a live point moving along the V-shaped error plot. Switch between forward, central and complex-step formulas and switch the precision between float16, float32 and float64 to see the bottom of the V move.

Try it: Halving h in three precisions

Integrator shoot-out *(in: Solving differential equations: Euler, RK4 and energy drift)*

Animate three pendulums side by side, one per integrator, with a live energy trace underneath. Sliders control h and the release angle. Then switch to a planet orbiting a star and watch Euler's orbit spiral outward while the symplectic orbit precesses but stays closed.

Try it: Open the integrator shoot-out

Runge's wiggle *(in: Interpolation: filling in between the points)*

Drag the number of nodes from 2 to 25 for $1/(1 + 25x^2)$ and toggle between equally spaced and Chebyshev nodes, and between a global polynomial and a cubic spline. Then drag individual data points and see how far the disturbance spreads: everywhere for the global polynomial, only locally for the spline.

Try it: Open Runge's wiggle · Compare a cubic spline

Version 0.1
Last updated 2026-10-14 13:05:05 -0700