Appendix A: The Mathematics You Need

Before You Start

Who this is for. A software engineer who took one calculus course and one probability course some years ago, has used neither since, and wants the minimum this book leans on. You write loops, records and tables for a living. Everything in this appendix is built from one of those three.

What it is not. A mathematics course. Nothing here is proved. Where a result is stated – that a smaller step gives a smaller error, that a sum of many small independent errors is bell-shaped – it is shown on numbers you can check with a calculator, and the Further Reading section says where the proof lives. If you want to know *why* a rule holds rather than *what* it does, this is the wrong document, and it says so rather than pretending otherwise.

How to use it. Each chapter's Before You Start names the sections of this appendix it assumes. Read those and skip the rest. The Where Each Section Is Used section at the end is the same map read backwards: from a section here to the chapters that lean on it, so that a reader who arrives from one pointer can see what else the same idea unlocks. One worked instance – a cup of coffee cooling on a desk – runs through every section, so a reader who has followed one section already knows the example the next one starts from. The body of the appendix names no chapter; the closing table does that once, for all of them.

One convention, stated once. Every displayed equation in this book is one line. A rule that needs two relations writes them on one line separated by a comma; a matrix is written row by row with semicolons between the rows. This is the book's mapped-mathematics convention, and it is what lets the same equation read as plain text in the Markdown source and as typeset mathematics in the printed page. Symbols are written the way you would type them: $*$ is multiplication, / is division, ^ is a power, _ is a subscript, and dT/dt is a derivative. The Notation section collects all of it in one table.

Pseudocode. Wherever a piece of mathematics is something a program does, it is also written as a short listing in an invented, Python-like pseudocode: a loop, a record, two nested loops, a priority queue. The listings are the mathematics restated for a reader who thinks in code; none is meant to run as written.

Learning objectives. By the end of this appendix you will be able to:

1. **Read** a differential equation aloud, **name** its state, its parameters and its initial condition, and **compute** the rate at a given state.
2. **Write** a differential equation in its integral form, **solve** the cooling equation in closed form, and **step** any equation forward by hand with Euler's method and one better method, **saying** what a smaller step buys and costs.
3. **Multiply** a matrix by a vector as two loops, **transpose** a matrix, **invert** a two-by-two, **write** a system of differential equations in state-space form, and **read** a mode and its natural frequency off a small system.
4. **Compute** a probability from counts, **apply** conditional probability and Bayes' rule with numbers, **test** two events for independence, and **read** a uniform and a Gaussian density for a probability, a mean and a standard deviation.
5. **Write** a Markov chain's transition matrix, **predict** a belief one step ahead, and **correct** it against a noisy reading.

6. **Compute** the cost of a simulation run and its real-time factor, and **decide** from that number whether a model can live inside a twin.

The exercises at the end name the objective each one serves.

Quantities, Rates and Equations

A quantity with a history

A temperature sensor on a cup of coffee reports a number every second. Written down, the numbers are a column in a table: a time and a value on each row. In code the value is a variable, T, and the table is its history. Mathematics writes the same thing as $T(t)$ – the temperature *at* time t – and drops the bracket, writing T, whenever the moment does not matter. Both spellings appear in this book; they name the same variable. A **quantity**, in this appendix, is a number that changes over time, and its history is the table.

Two other quantities appear in the same story. The room is at T_a, which does not change over the ten minutes we care about, and the coffee starts at $T(0) = 80$ degrees. A quantity that does not change over the time span you are modelling is a **constant** for that model, even if it is a variable in the world.

The rate: how fast it is changing now

Look at two neighbouring rows of the table. The coffee read 80.0 at minute zero and 74.3 at minute one. It lost 5.7 degrees in one minute, so its **rate of change** over that minute was about -5.7 degrees per minute. Take two rows closer together – a second apart – and the rate over that second is a little steeper, closer to -6.0, because the coffee cools fastest when it is hottest. The **derivative** is the rate you would read from two rows infinitely close together: how fast the quantity is changing *right now*, not on average over some interval. It is written dT/dt and read aloud as "d T d t" or, more usefully, "the rate of T". The units are the quantity's units per time unit: degrees per minute here.

Three things about the rate that a programmer may not have been told:

- **It is a quantity too.** dT/dt has a value at every moment and a history of its own. It can be plotted; it can be measured, roughly, as the difference of two rows divided by the time between them.
- **Its sign is direction.** Negative means falling, positive rising, zero means the quantity is, for that moment, not moving.
- **It is local.** The rate at minute three says nothing about minute four. What connects the two is an equation.

Ordinary Differential Equations

What they are for. Most of what an engineer knows about a physical system is not a formula for its future but a statement about how it changes: heat flows from hot to cold in proportion to the difference; water leaves a tank in proportion to the pressure at the outlet; a population grows in proportion to its size. Each of those is a rule for the *rate* of a quantity in terms of the quantity's present value. An **Ordinary Differential Equation** (ODE) is exactly that: an equation in which the unknown is a quantity's whole history and the equation relates the quantity's rate to its present value and a few constants. "Ordinary"

because there is one independent variable, time; "differential" because a derivative appears in it. The equation is the model; the history it implies is what a simulator computes, and what a twin compares against the sensor.

Newton's cooling law as the example. Newton observed that a warm object loses heat in proportion to how much warmer it is than its surroundings. As an equation for the rate:

$$\frac{dT}{dt} = -k\,(T - T_{\mathrm{a}}) \tag{1}$$

Read Equation 1 aloud: "the rate of T is minus k times the excess of T over the room". Everything an ODE has is on show. The **state** is T, the one quantity whose history the equation fixes. The **parameters** are the constants in the rule: k, in units of one over minutes, which says how quickly this cup in this room loses heat, and T_{a}, the room temperature. Change the cup for a vacuum flask and k falls; move the cup outside and T_{a} falls. Neither is a quantity the equation moves; both are numbers the equation uses. When this book speaks of parameters collectively it writes θ.

An equation alone does not say where the coffee is. It says how fast it is moving *given where it is*. To fix an actual history you also need to know where it started: the **initial condition** $T(0) = 80$. The pair – equation plus initial condition – fixes the whole future history, the **trajectory**, and nothing less does. Two cups obeying the same equation from different starting temperatures follow different trajectories; two cups from the same starting temperature with different k do too.

Order, linearity, and the words you will meet. The cooling law is **first-order**, because only the first derivative appears; a vibrating structure obeys a **second-order** equation, in which the acceleration d^2x/dt^2 appears, and the Many Quantities section shows how a second-order equation is rewritten as two first-order ones. The cooling law is also **linear**: the rate is a weighted sum of the state and a constant, with no square, product or function of the state in it. A draining tank, whose outflow goes with the square root of the level, is **non-linear**. Linear equations are the ones with formulas for their solutions and the ones a matrix can hold; non-linear ones are the ordinary case in the world and are handled by the loop of the next two sections.

Difference equations. Every worked example in this appendix ends up computing the trajectory step by step, and the rule a step follows has its own name. A **difference equation** gives the next value of a quantity from its present value, indexed by step rather than by time:

$$T_{k+1} = F(T_k) \tag{2}$$

In Equation 2 the subscript k counts steps, so T_k is the value after k of them and T_0 is the initial condition. A difference equation is the discrete cousin of a differential equation: the same idea of "given where you are, here is where you go", with the step of time built in rather than shrunk to nothing. Some systems are difference equations to begin with – a bank balance compounding monthly, a counter incremented per event, a sensor that reports once per interval – and a differential equation becomes one the moment a program steps it, which is the subject of the From an Equation to a Loop section. The pair appears everywhere in this book: a differential equation is how the physics is written down, and a difference equation is what the twin actually runs.

Worked example 1: reading the equation at three temperatures

Take $k = 0.1$ per minute and $T_\mathrm{a} = 20$ degrees. At $T = 80$, the excess over the room is 60 degrees, so the rate is $-0.1 \times 60 = -6$ degrees per minute. The reasoning behind each move: subtract the room temperature first, because the equation is about the *excess* and not the temperature; then multiply by k, whose units of one over minutes turn degrees into degrees per minute; then attach the sign, which says the coffee is cooling.

At $T = 50$ the excess is 30 and the rate is $-0.1 \times 30 = -3$: half the excess, half the rate. That proportionality is the whole content of the equation.

At $T = 20$ the excess is zero and so is the rate. The coffee has reached the room and stops moving. A value at which the rate is zero is an **equilibrium**, and this equation has exactly one.

Worked example 2: the same equation, a different parameter

Keep the coffee at $T = 80$ and change the cup. A thin paper cup has $k = 0.2$; the rate at 80 becomes $-0.2 \times 60 = -12$ degrees per minute, twice as fast. Now keep the paper cup and carry it into a warmer room at $T_\mathrm{a} = 30$: the excess is 50 and the rate is $-0.2 \times 50 = -10$. Same state, same equation, different parameters, different rate. When a model is "kept current" by re-estimating its parameters, this is what moves: not the equation, and not the state, but k and T_a.

Worked example 3: a difference equation from a watering rig

A potted plant on a bench has a soil-moisture probe read every ten seconds, and between waterings the soil dries at a steady $d = 1.0$ percentage points per hour – a number read off two probe readings eight hours apart, $d = (44 - 36)/8 = 1.0$. As a differential equation the drying is the simplest kind, a constant rate:

$$\frac{dm}{dt} = -d$$

State m, the soil moisture; parameter d; initial condition $m(0) = 44$. Because the rate does not depend on m, the trajectory is a straight line – m falls one point every hour. As a difference equation stepped once per probe reading, with the reading period $T_\mathrm{r} = 10$ seconds written in hours:

$$m_{k+1} = m_k - d\,T_\mathrm{r}$$

Ten seconds is 1/360 of an hour, so each step lowers m by $1.0/360 = 0.0028$ points, and 360 steps later, one hour on, m has fallen by one point as it should. The rig's controller was running a difference equation all along; the differential equation is its statement of the physics, and the difference equation is what the code computes. Because the rate here does not depend on the state, the two agree exactly; the next two sections are about what happens when it does.

The Integration

From a rate to a total

A differential equation says how fast a quantity is moving. To know where the quantity *is*, you have to add up all that movement from the start until now. Adding up a rate over time is **integration** – the area under the rate's curve – and it is the second half of the calculus course you took. If the rate of T is $f(T)$, then the temperature at time t is the temperature at the start plus the accumulated rate:

$$T(t) = T(0) + \int_0^t f\big(T(s)\big)\, ds \tag{3}$$

Equation 3 is the **integral form** of the differential equation, and it says the same thing the differential form says, read the other way round. The differential form asks "how fast, right now?"; the integral form asks "how much, so far?". The variable s inside the integral is a stand-in for every moment between zero and t, and ds is the width of the sliver of time each rate value applies to. Nothing has been solved yet: T appears on both sides, inside the integral and out. But the form is what every solver in the next section is approximating, and it is what a closed-form solution, where one exists, comes from.

Solving the cooling equation in closed form

For the cooling law the integral can be done by hand, because the equation is linear. The steps are the ones your calculus course drilled, and each is shown so the result can be checked rather than believed. Write the excess over the room as $y = T - T_\mathrm{a}$; since T_a is constant, $dy/dt = dT/dt$, and the equation becomes $dy/dt = -k\,y$. Separate the two variables, putting everything in y on one side and everything in t on the other: $dy/y = -k dt$. Integrate both sides from the start to time t: the left side gives $\ln(y(t)) - \ln(y(0))$, the right gives $-k\,t$. Undo the logarithm with the exponential, and put T back:

$$T(t) = T_\mathrm{a} + \big(T(0) - T_\mathrm{a}\big)\, e^{-k\,t} \tag{4}$$

Equation 4 is the **closed-form solution**: a formula that gives the temperature at any time directly, with no stepping. The excess over the room decays exponentially, and $1/k$ – ten minutes for the coffee – is the **time constant**, the time in which the excess falls to about 37 per cent of what it was. Every cooling curve drawn in this appendix, and every "exact" value the worked examples compare a loop against, comes from this line.

Why the loop is still needed. The trick above – separate the variables, integrate each side – works because the equation is linear and has one state. Add a second coupled quantity with a non-linear exchange between them, or a rate that depends on a measured input switching on and off, and there is no formula. For almost every equation a twin uses, the integral form is exact and unsolvable by hand, and the trajectory is produced by approximating the integral one sliver of time at a time. That is the loop, and it is why an equation with no closed form is no obstacle to a twin.

Worked example 1: the closed form at three minutes

Check the formula against the numbers the rest of the appendix uses: $k = 0.1$, $T_\text{a} = 20$, $T(0) = 80$. At $t = 1$, $\exp(-0.1) = 0.9048$, so $T(1) = 20 + 60 \times 0.9048 = 74.29$. At $t = 3$, $\exp(-0.3) = 0.7408$, so $T(3) = 20 + 60 \times 0.7408 = 64.45$. At $t = 10$, one time constant, $\exp(-1) = 0.3679$, so $T(10) = 20 + 60 \times 0.3679 = 42.07$: the excess of 60 has fallen to 22.07, which is 36.8 per cent of it, as the time constant promised.

Worked example 2: a question the closed form answers directly

When does the coffee reach 30 degrees? The loop would have to run until it got there; the formula is solved for t. Set $T(t) = 30$, so the excess is 10 out of an initial 60: $\exp(-0.1\,t) = 10/60$. Take logarithms: $-0.1\,t = \ln(1/6) = -1.792$, so $t = 17.9$ minutes. Questions of the form "when will it reach", "how long until", and "what is it at exactly this time" are where a closed form earns its keep, and they are the questions a monitoring dashboard asks.

Worked example 3: the integral form for the drying soil

The drying equation $\frac{dm}{dt} = -d$ has a rate that does not depend on the state, so its integral form can be integrated at once: $m(t) = m(0) + \int_0^t (-d)\,ds = m(0) - d\,t$. With $m(0) = 44$ and $d = 1.0$, $m(8) = 44 - 8 = 36$, which is the reading the drying rate was computed from in the first place. The lesson is the one the closed form of the cooling law taught from the other side: the integral form is always right, and the only question is whether the integral can be done by hand. Here it can, because the integrand is a constant; for the coffee it could, because the equation is linear; for the coupled coffee and mug of the Many Quantities section it still can, with more algebra; for a draining tank with a square root in it, it cannot, and the loop takes over.

From an Equation to a Loop

What a solver is

A **solver** is the program that turns a differential equation into a trajectory. It holds the state, chooses a step of time, uses the equation to move the state across that step, records the result, and repeats until the end of the run. Every solver is an approximation of the integral form of the previous section: the integral adds up the rate over every instant, and a solver adds it up over a finite number of slivers instead. The methods differ only in how they estimate the rate across one sliver. This section builds the simplest one by hand and then shows two better ones at the same level of detail.

Euler's step: the integral over one sliver

Take the integral form of the previous section over one short sliver of time, from t to $t + h$. The exact change in T is the area under the rate curve across that sliver. The crudest estimate of that area is a rectangle: the rate at the *start* of the sliver, held constant across its width h. Rate times width is the rectangle's area, and adding it to the starting value gives **Euler's step**:

$$T_{k+1} = T_k + h\,f(T_k) \tag{5}$$

New value equals old value plus rate times step. In Equation 5, h is the **step size**, the time one step advances; $f(T_k)$ is the equation's right-hand side evaluated at the present value, $-k\,(T_k - T_\mathrm{a})$ for the coffee; and the subscript k counts steps. The step counter k and the cooling parameter k are different letters in the printed mathematics and the same letter in plain text; this appendix writes the parameter as k and the step in the subscript position only, and the Notation section says how the book keeps them apart. Euler's step is a difference equation – the previous section's $F(T_k)$ with $T_k + h\,f(T_k)$ filled in – which is the sense in which the loop *is* a difference equation manufactured from a differential one.

As a loop:

```
T      = 80.0          # initial condition
T_a    = 20.0          # parameter
k      = 0.1           # parameter, per minute
h      = 1.0           # step size, minutes
t      = 0.0
while t < 30.0:
    rate = -k * (T - T_a)     # the right-hand side, at the present value
    T    = T + h * rate       # Euler's step: one rectangle of the
 ↪  integral
    t    = t + h
    record(t, T)
```

Ten lines, and every solver of a differential equation in this book is this loop with a better line eight. The loop is what "integrating the equation" means to a program: adding up rate times step, one step at a time, from the initial condition forward. There is nothing else to it.

Worked example 1: three steps by hand

Step size $h = 1$ minute, $k = 0.1$, $T_\mathrm{a} = 20$, $T_0 = 80$.

- Step 1. Rate at 80 is $-0.1 \times 60 = -6$. New value $T_1 = 80 + 1 \times (-6) = 74$.
- Step 2. Rate at 74 is $-0.1 \times 54 = -5.4$. New value $T_2 = 74 - 5.4 = 68.6$.
- Step 3. Rate at 68.6 is $-0.1 \times 48.6 = -4.86$. New value $T_3 = 68.6 - 4.86 = 63.74$.

The closed form gives 74.29, 69.12 and 64.45 at those three minutes, so the loop is under the truth by 0.29, 0.52 and 0.71 degrees. The error grows with each step and stays small; both facts are typical. The reason the loop undershoots is visible in the rectangle: it holds the rate at its *starting* value for the whole sliver, the true rate eases during the sliver, so every rectangle is a little too tall and the loop cools the coffee a little too fast every minute.

Worked example 2: the same three minutes with a larger step

Now take $h = 5$ minutes and do one step. Rate at 80 is -6; new value $T_1 = 80 + 5 \times (-6) = 50$. The closed form gives 56.39 at minute five, so the error is 6.4 degrees – twenty times the error of the one-minute step at minute one, for a step five times longer, because a wide rectangle misses much more of the curve's bend than a narrow one. Take $h = 10$ and one step lands on $80 - 60 = 20$, the room temperature, where the exact curve reads 42.07. Take

$h = 25$ and one step gives $80 + 25 \times (-6) = -70$: the coffee has overshot the room by ninety degrees and would, on the next step, shoot back above it. The loop has not merely become inaccurate; it has become unstable, and it does so for this equation whenever $h > 2/k$, here twenty minutes.

Figure 1 draws the three runs beside the exact curve.

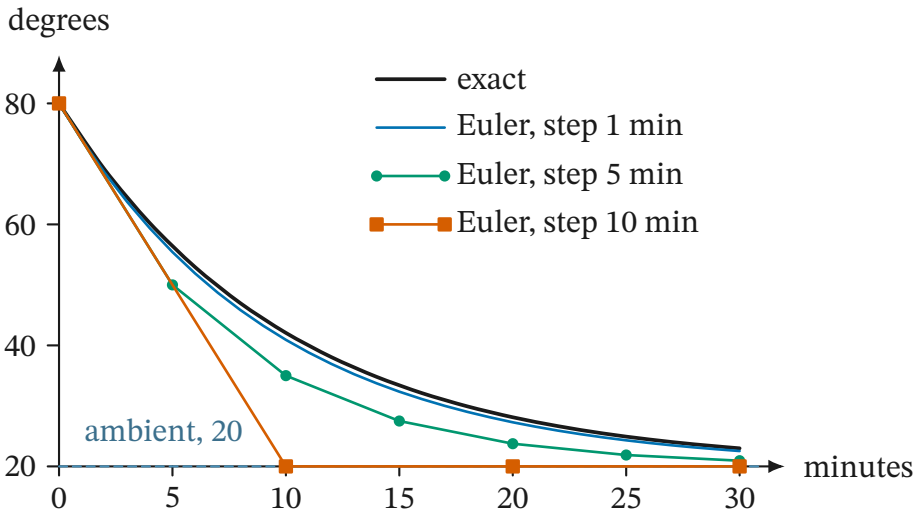


Figure 1: The cooling curve from the same equation, initial condition and parameters, computed with Euler's step at three step sizes, beside the closed-form trajectory.

At a one-minute step the loop's curve lies under the exact one by less than a degree everywhere. At five minutes it visibly undershoots and then approaches the room from below. At ten minutes it falls to the room in one step and stays there: right about where the coffee ends up, wrong about everything in between. **What the step size buys and costs**, in one sentence: a smaller h gives a trajectory closer to the truth, and a run of the same length takes proportionally more steps. Halve the step and you double the work; for Euler's method you also roughly halve the error at the end of the run.

Worked example 3: sampled time, with the sensor's period as the step

A twin stepping the coffee equation does not get to choose its clock freely: the thermometer reports every T_s seconds, and the twin's prediction is compared against each reading as it arrives. The natural choice is to make the sensor's period the step, $h = T_s$, so that one loop iteration produces one predicted value for each reported one. With a thermometer reporting every 30 seconds, $h = 0.5$ minutes, and the first two steps are $T_1 = 80 + 0.5 \times (-6) = 77$, then the rate at 77 is -5.7 and $T_2 = 77 - 2.85 = 74.15$, against a closed-form 74.29 at minute one: an error of 0.14, half the one-minute step's, as the step-size rule predicts. A loop run this way is in **sampled time**: one step per reading, so that the predicted value and the reported value arrive on the same clock and can be compared, which is the comparison a state estimator makes at every step.

Better Solvers

Euler's step is the simplest member of a large family, and the two members below are each one idea away from it. Both are shown at the same level of detail as Euler's step, on the

same coffee, so the improvement is a number rather than a claim.

Heun's step: average the rate at both ends. The rectangle's fault is that it uses the rate at the start of the sliver only. A better estimate of the area is a trapezium: the average of the rate at the start and the rate at the end. The end rate is not known before the step is taken, so Heun's method takes an Euler step as a *guess* at the end value, reads the rate there, and averages the two:

$$T_{\text{guess}} = T_k + h\,f(T_k), \quad T_{k+1} = T_k + \frac{h}{2}\big(f(T_k) + f(T_{\text{guess}})\big) \tag{6}$$

Two rate evaluations per step instead of one, and in Equation 6 the error at the end of a run shrinks with the *square* of the step rather than in proportion to it, so halving the step quarters the error. For the coffee with $h = 1$: the start rate is -6, the guess is 74, the rate there is -5.4, and $T_1 = 80 + 0.5 \times (-6 - 5.4) = 74.3$, against a closed form of 74.29 – an error of 0.01 where Euler's was 0.29. With $h = 5$: start rate -6, guess 50, rate there -3, $T_1 = 80 + 2.5 \times (-6 - 3) = 57.5$, against 56.39, where Euler gave 50. This is the two-stage member of the **Runge-Kutta** family; the widely used four-stage member samples the rate four times per step and its error shrinks with the fourth power of the step, so a step ten times longer gives the same accuracy at a quarter of the rate evaluations. The whole family exists because the fixed-step rectangle is expensive for the accuracy it buys [1].

```
rate1 = f(T)                    # rate at the start
guess = T + h * rate1           # an Euler step, used only to look ahead
rate2 = f(guess)                # rate at the guessed end
T     = T + h * (rate1 + rate2) / 2
```

The implicit step: use the rate at the end. Euler's step uses the rate at the start; the **implicit** (or backward) Euler step uses the rate at the *end* of the sliver, at the value it is about to compute:

$$T_{k+1} = T_k + h\,f(T_{k+1}) \tag{7}$$

The unknown T_{k+1} appears on both sides of Equation 7, so each step is a small equation to solve rather than a formula to apply. For the cooling law it solves in one line: $T_{k+1} = T_k - h\,k\,(T_{k+1} - T_a)$ rearranges to $T_{k+1} = (T_k + h\,k\,T_a)/(1 + h\,k)$. For $h = 5$: $T_1 = (80 + 0.5 \times 20)/1.5 = 60$, against a closed form of 56.39 – less accurate than Heun, more accurate than Euler's 50, and, the reason it exists, *stable at any step size*. For $h = 25$, where Euler's loop overshot to -70, the implicit step gives $(80 + 2.5 \times 20)/3.5 = 37.1$: wrong by fifteen degrees, but on the right side of the room and heading the right way. For a non-linear equation the small equation is solved numerically at every step, which is the price; the reward is that the step can be as long as the accuracy allows rather than as short as stability demands.

Adaptive steps. A production solver does not use one h for the whole run. It takes a step, estimates its own error – Heun's method gives one for free, as the difference between the guess and the corrected value – and shortens the step where the curve bends or something switches, lengthening it again where the curve is gentle. Solvers of this kind take long steps while the dynamics are slow and short ones where they are fast or a switching instant must be located [2]. The user gives a tolerance instead of a step, and the solver spends its effort where the equation needs it.

Every simulator a twin might use offers these methods. This appendix uses Euler's step because it is the one you can do by hand, and every other solver is Euler with a better line eight. What none of them can remove is the cost of an equation that is *stiff*, which the Many Quantities section explains once the second quantity that causes it is on the table.

Many Quantities at Once

A twin rarely tracks one number. The coffee and the mug it sits in have two temperatures; a wind turbine's tower has a position and a velocity at every point along it; a battery has a charge, a temperature and an internal resistance. This section builds, one piece at a time, the notation that lets many quantities be handled as one: the vector, the matrix, the one operation between them, and then the system of differential equations they make possible.

Vectors: a record whose fields are numbers

Mathematics bundles the quantities of one system into a **vector**, and a vector is nothing more than a record whose fields are all numbers and are listed in a fixed order:

$$x = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}$$

The semicolon separates the fields, which mathematics stacks vertically and calls **components**; x is the whole record. The number of fields is the vector's **dimension**, n, and this book reserves x for the record of everything a model must remember – its **state** – so x has n components whenever the state has n quantities. The book sets a vector in the same plain letter as a single quantity; whether x has one component or a thousand is said in words where it matters. Two vectors of the same dimension are added field by field, and a vector is multiplied by a number by multiplying every field: $2\left[\begin{smallmatrix} 3 \\ 4 \end{smallmatrix}\right]$ is $\left[\begin{smallmatrix} 6 \\ 8 \end{smallmatrix}\right]$. Those two operations, and nothing else, are what "vector arithmetic" means in this book.

```
x = [T_1, T_2]              # a record: two named fields, in a fixed
 ↪ order
y = [a + b for (a, b) in zip(x, w)]     # addition, field by field
z = [2 * a for a in x]                  # a number times a vector
```

Matrices: a table of coefficients, and two things done to it

A **matrix** is a table of coefficients, m rows by n columns, and A_{ij} names the entry in row i and column j. Written on one line, rows are separated by semicolons and entries within a row by commas:

$$A = \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}$$

is the two-by-two table whose first row is $2, 1$ and whose second is $0, 3$. A matrix with as many rows as columns is **square**, and the square matrix with ones down the diagonal and zeros elsewhere is the **identity**, I, which leaves every vector unchanged. Two things are done to a matrix in this book, and both are mechanical.

The transpose A^{T} is the table with rows and columns swapped: A_{ij} moves to row j, column i. For $A = \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}$, $A^{\mathsf{T}} = \left[\begin{smallmatrix} 2 & 0 \\ 1 & 3 \end{smallmatrix}\right]$. A matrix equal to its own transpose is **symmetric**; every covariance matrix in the Probability section is.

The inverse A^{-1} is the table that undoes A: apply A and then A^{-1} and every vector comes back unchanged, $A^{-1} A\, x = x$. For a two-by-two there is a formula and the book needs no larger case:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{a\,d - b\,c} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{8}$$

Swap the diagonal, negate the off-diagonal, divide by $a\,d - b\,c$. That divisor is the **determinant**, and when it is zero the inverse in Equation 8 does not exist: the two rows of the table say the same thing in different words, so no table can tell their outputs apart. An inverse is what solving a linear system of equations means – if $A\,x = b$ then $x = A^{-1}\,b$ – and a zero determinant is the algebraic form of "these equations do not fix a unique answer".

Worked example 1: a transpose and an inverse, checked

$A = \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}$. The transpose is $A^{\mathsf{T}} = \left[\begin{smallmatrix} 2 & 0 \\ 1 & 3 \end{smallmatrix}\right]$. The determinant is $2 \times 3 - 1 \times 0 = 6$. Swap the diagonal to get 3 and 2, negate the off-diagonal to get -1 and -0, divide by six: $A^{-1} = \left[\begin{smallmatrix} 0.5 & -0.1667 \\ 0 & 0.3333 \end{smallmatrix}\right]$. The check is in the next worked example, once multiplication is defined; the lesson here is that both operations are pure table manipulation, with no judgement in them. Now try $\left[\begin{smallmatrix} 1 & 2 \\ 2 & 4 \end{smallmatrix}\right]$: the determinant is $1 \times 4 - 2 \times 2 = 0$, the second row is twice the first, and no inverse exists.

Multiplication: each output is a weighted sum of the inputs

The one operation this book needs between a matrix and a vector is $y = A\,x$. Its meaning fits in a sentence: **each output component is a weighted sum of the input components, and row i of the table holds the weights for output i.** As code, that is two nested loops:

```
for i in 0 .. m-1:            # one output per row
    y[i] = 0
    for j in 0 .. n-1:        # weighted sum across the row
        y[i] = y[i] + A[i][j] * x[j]
```

If you have ever computed a weighted average, or a dot product, or a dense layer of a neural network, you have run this loop. Two matrices are multiplied the same way, one column of the second at a time, and that is the only other product this book uses: $A\,B$ is the table whose column j is A applied to column j of B. Order matters – $A\,B$ and $B\,A$ are different tables in general – which is why every equation in this appendix is careful about which side a matrix sits on.

Worked example 2: a two-by-two times a vector, and the inverse checked

$A = \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}$ and $x = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$. Output component one is row one against x: $2 \times 4 + 1 \times 5 = 13$. Output component two is row two against x: $0 \times 4 + 3 \times 5 = 15$. So $y = \begin{bmatrix} 13 \\ 15 \end{bmatrix}$. Trace the loop: $i = 0$ runs j over 0 and 1 and accumulates 8 then 13; $i = 1$ accumulates 0 then 15. Four multiplications and two additions, which is n^2 multiplications for a square table – a number to remember when the section turns to cost.

Now the check the previous example promised. Apply $A^{-1} = \begin{bmatrix} 0.5 & -0.1667 \\ 0 & 0.3333 \end{bmatrix}$ to $y = \begin{bmatrix} 13 \\ 15 \end{bmatrix}$: row one gives $0.5 \times 13 - 0.1667 \times 15 = 6.5 - 2.5 = 4$, row two gives $0 \times 13 + 0.3333 \times 15 = 5$. The original $x = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$ comes back, which is what "undoes" means.

Systems of Differential Equations

Two equations, each mentioning the other's variable. Put the coffee in a heavy ceramic mug. Now the coffee, at T_1, loses heat to the mug, at T_2, and the mug loses heat to the room. There are two quantities and two equations, and neither equation can be solved on its own, because each one's rate depends on the other one's value:

$$\frac{dT_1}{dt} = -k_1 (T_1 - T_2), \quad \frac{dT_2}{dt} = k_2 (T_1 - T_2) - k_3 (T_2 - T_a)$$

The first says the coffee cools in proportion to how much hotter it is than the mug. The second says the mug warms in proportion to the same difference, with its own constant k_2 because a mug's heat capacity is not a coffee's, and cools to the room through k_3. Two equations that share variables are a **system of differential equations**, and a system is the ordinary case: almost nothing in a real asset changes on its own.

Excess over the room removes the constant. Measure both temperatures as excess over the room, $y_1 = T_1 - T_a$ and $y_2 = T_2 - T_a$. Since T_a is constant the rates are unchanged, and the room drops out of both equations: $dy_1/dt = -k_1 (y_1 - y_2)$ and $dy_2/dt = k_2 (y_1 - y_2) - k_3 * y_2$. Now every rate is a weighted sum of the two present values and nothing else – a **linear system**.

One matrix holds all the weights. Collect the two excesses into a vector $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ and the weights into a table, row one for the coffee's rate and row two for the mug's, and the pair of equations is one line:

$$\frac{dy}{dt} = A\,y, \quad A = \begin{bmatrix} -k_1 & k_1 \\ k_2 & -(k_2 + k_3) \end{bmatrix} \tag{9}$$

Equation 9 is the one-quantity cooling law with a matrix where $-k$ used to be, and the derivative of a vector is the vector of derivatives, component by component. Row one of A multiplied into y gives $-k_1\,y_1 + k_1\,y_2$, which is the coffee's equation; row two gives the mug's. Nothing has been added and nothing lost; the matrix is a filing system for the coefficients.

Euler's step for a vector. The loop of the previous section applies unchanged, with the multiplication above in place of the scalar product: $y_{k+1} = y_k + h\,A\,y_k$. One matrix-times-vector per step – the two nested loops – inside the one loop that steps time.

```
y = [60.0, 0.0]                          # coffee excess, mug excess
A = [[-0.1, 0.1], [0.3, -0.5]]
h = 1.0
while t < 30.0:
    rate = matvec(A, y)                  # the two nested loops
    y    = [y[i] + h * rate[i] for i in 0 .. 1]
    t    = t + h
```

A second-order equation is a two-equation system. A vibrating mass obeys $d^2x/dt^2 = -(k_s/m_s)\,x$: its acceleration pulls it back in proportion to its displacement. Introduce the velocity $v = dx/dt$ as a second state, and the one second-order equation becomes two first-order ones, $dx/dt = v$ and $dv/dt = -(k_s/m_s)\,x$, which is the system $\frac{d}{dt}\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right] = \left[\begin{smallmatrix} 0 & 1 \\ -(k_s/m_s) & 0 \end{smallmatrix}\right]\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]$. Every higher-order equation is handled this way, which is why a solver needs only first-order systems and why "state" so often means position *and* velocity.

Worked example 3: two Euler steps of the coupled system

Take $k_1 = 0.1$, $k_2 = 0.3$, $k_3 = 0.2$ per minute, so $A = \left[\begin{smallmatrix} -0.1 & 0.1 \\ 0.3 & -0.5 \end{smallmatrix}\right]$, and start with the coffee at 80 and the mug at room temperature: $y_0 = \left[\begin{smallmatrix} 60 \\ 0 \end{smallmatrix}\right]$.

- Step 1. The rate is $A\,y_0$: row one gives $-0.1 \times 60 + 0.1 \times 0 = -6$, row two gives $0.3 \times 60 - 0.5 \times 0 = 18$. The coffee is cooling at 6 degrees a minute and the mug warming at 18, fast, because a cold mug against hot coffee is a large excess through a large k_2. With $h = 1$, $y_1 = \left[\begin{smallmatrix} 60-6 \\ 0+18 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 54 \\ 18 \end{smallmatrix}\right]$: coffee 74, mug 38.
- Step 2. The rate at y_1 is row one $-0.1\times54+0.1\times18 = -3.6$, row two $0.3\times54-0.5\times18 = 7.2$. So $y_2 = \left[\begin{smallmatrix} 54-3.6 \\ 18+7.2 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 50.4 \\ 25.2 \end{smallmatrix}\right]$: coffee 70.4, mug 45.2.

The coffee's rate fell from -6 to -3.6 in one minute, far faster than the lone cup's did, because the mug it is losing heat to is catching up with it; the mug's rate fell from 18 to 7.2 for the same reason. The reasoning behind each move is the same as in the one-quantity case, done once per row: evaluate the equation at the present record, multiply by the step, add.

State-Space Equations

The form. A system of differential equations becomes a **state-space model** when three more things are named: an **input** u, the quantities done to the system from outside – a heater's power, a valve setting, a command – an **output** z, the quantities a sensor actually reports, and the tables that say how each enters. For a linear system the whole model is two lines on one:

$$\frac{dx}{dt} = A\,x + B\,u, \quad z = C\,x \tag{10}$$

In Equation 10, A is the table of the system's own dynamics, as in the coupled cooling; B says which states an input pushes on and how hard; C says which combination of states a sensor sees. For the coffee and mug with a small heater under the mug delivering u watts, and a thermometer in the coffee only: $x = \left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right]$, $B = \left[\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right]$ where b is the degrees per minute one watt buys the mug, and $C = \left[\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right]$, because the thermometer reads the first state and nothing of the second. A non-linear system uses the same shape with functions in place of

the tables, $dx/dt = f(x, u, \theta)$ and $z = g(x)$, and the Markov section adds the noise terms that turn this into the model a state estimator uses.

The discrete form. A twin runs the state-space model as a difference equation, in sampled time. Euler's step with h equal to the sensor period gives it directly:

$$x_{k+1} = F\,x_k + G\,u_k, \quad z_k = C\,x_k, \quad F = I + hA, \quad G = hB \tag{11}$$

Equation 11 is the **discrete-time state-space model**, and every symbol in it is a table computed once before the loop starts. The step from the continuous form to this one is called **discretisation**, and it is what turns a physical model into a process that produces a sample per period [3]. Euler's $F = I + hA$ is the simplest discretisation; a solver's better step gives a better F, and for a linear system there is an exact one.

What the form is for. Three things, and all three run through this book. First, *uniformity*: a solver, an estimator or a controller written against (A, B, C) works for any linear system, whatever the physics, which is what lets one estimator serve a coffee cup and a turbine tower. Second, *the estimator itself*: a state estimator needs to know how the state moves on its own, how the input moves it, and what the sensor would report for a given state, and those are exactly A, B and C – a physics-based twin of a floating wind turbine, for instance, runs a Kalman filter over a linear model of the structure to estimate the structural states and the section loads along the tower [4]. Third, *the unmeasured*: C is usually a short, wide table, because a sensor sees a few combinations of many states, and the whole point of carrying x rather than z is to reason about the states the sensor does not see.

```
F = I + h * A                  # computed once, before the loop
G = h * B
x = x_0
for each sensor period k:
    x = matvec(F, x) + matvec(G, u[k])     # predicted state
    z = matvec(C, x)                       # what the sensor should
 ↪  report
    record(k, x, z)
```

Worked example 4: the coffee and mug in state-space form

With $h = 1$ minute and $A = \left[\begin{smallmatrix} -0.1 & 0.1 \\ 0.3 & -0.5 \end{smallmatrix}\right]$, $F = I + hA = \left[\begin{smallmatrix} 0.9 & 0.1 \\ 0.3 & 0.5 \end{smallmatrix}\right]$. Apply F to $x_0 = \left[\begin{smallmatrix} 60 \\ 0 \end{smallmatrix}\right]$: row one $0.9 \times 60 + 0.1 \times 0 = 54$, row two $0.3 \times 60 + 0.5 \times 0 = 18$, so $x_1 = \left[\begin{smallmatrix} 54 \\ 18 \end{smallmatrix}\right]$ – the same answer as Euler's step in the previous worked example, because F *is* Euler's step written as a table. Now switch the heater on at $u = 20$ watts with $b = 0.05$ degrees per minute per watt: $G = hB = \left[\begin{smallmatrix} 0 \\ 0.05 \end{smallmatrix}\right]$, and $Gu = \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$, so $x_1 = \left[\begin{smallmatrix} 54 \\ 19 \end{smallmatrix}\right]$: the heater adds one degree a minute to the mug and nothing directly to the coffee. The output is $z_1 = C\,x_1 = 54$ – the thermometer reports the coffee's excess and knows nothing of the mug's, which a twin would then have to infer.

Stiff Equations for Digital Twins

The word **stiff** describes a system, not a solver, and it appears the moment a system has two quantities that move at very different speeds. Put a thermocouple in the coffee: the probe tip settles to the coffee's temperature with its own k of perhaps 100 per minute, while the coffee

itself cools with $k = 0.1$. Euler's step is stable for the probe only below 2/100, or about a second, so a fixed-step Euler loop must take a step every second for a whole hour to follow a probe that settled in the first three – a thousand times the work the coffee curve needs on its own account. A system is stiff when it holds a very fast process and a very slow one at once and the fast one is over almost immediately, so that stability, not accuracy, dictates the step.

The coupled coffee and mug are mildly stiff already: the Modes subsection below shows that the system has a fast part with a time constant of under two minutes and a slow part with one of nearly thirty, so Euler's step must stay under about three and a half minutes for the fast part while the slow part would be happy with ten. In a twin, stiffness is the ordinary case rather than the exception – an electrical transient inside a thermal process, a controller switching inside a slow plant, a probe inside a vessel – and it is the reason the implicit and adaptive solvers of the previous section exist. If a model of yours runs inexplicably slowly at a fixed step, stiffness is the first thing to suspect, and the remedy is an implicit step, which the previous section showed is stable at any step size, or a solver that adapts.

Dimension, and cost growing with it

The two loops cost $m\,n$ multiplications, n^2 for a square table. The coffee and mug have $n = 2$ and four multiplications a step. A model of a turbine tower that keeps a position and a velocity at five hundred points along it has $n = 1000$ and a million multiplications a step. A finite-element model of the same tower keeps three displacements at each of a few hundred thousand mesh points and has n around a million, and a million squared is 10^{12} multiplications a step. It is the same loop with a million-row table. The table is mostly zeros, because each mesh point touches only its neighbours, and every real simulator exploits that; the point stands that cost grows with dimension, and faster than dimension. The What a Solver and a Simulator Do section turns this into the one number that decides whether a model can live inside a twin.

Modes: the shapes a system prefers to move in

The idea. Pluck a guitar string and it vibrates in a shape: the whole string bowing out and back, at one pitch. Touch it lightly at the midpoint and pluck again: two half-length bows moving oppositely, at twice the pitch. Each shape, with its own frequency, is a **mode** of the string – a pattern of motion the structure prefers, in which every point moves in step at one **natural frequency**. Modal analysis of a monitored structure extracts exactly these from vibration recordings: the natural frequencies, which carry most of the structure's vibration energy, and the mode shapes, the spatial patterns it vibrates in at each of them [5].

The mathematics, for a linear system. A mode of $dy/dt = A\,y$ is a vector v that the matrix does not turn, only stretches: applying A to v gives back v times a number λ. If the system starts in such a shape it stays in it, every component scaled by the same $\exp(\lambda\,t)$:

$$A\,v = \lambda\,v, \quad y(t) = e^{\lambda t}\,v \tag{12}$$

The pair in Equation 12 is the mode shape v and its rate λ; a negative λ is a decay with time constant $-1/\lambda$, and for a vibrating system λ comes as a pair whose imaginary part is the natural frequency. Any starting state is a mix of the modes, and the trajectory is that mix with each mode decaying or oscillating at its own rate, which is why a handful of modes

describes most of what a large linear system does. Finding v and λ from A is a standard computation every numerical library provides; for a two-by-two it is a quadratic equation, and the worked example does it.

Natural frequency, for a vibrating mass. The single mass on a spring of the previous subsection obeys $d^2x/dt^2 = -(k_s/m_s)\,x$, and the motion $x(t) = a\,\cos(\omega\,t)$ satisfies it exactly when $\omega^2 = k_s/m_s$:

$$\omega = \sqrt{k_s/m_s}, \quad f_n = \frac{\omega}{2\pi} \tag{13}$$

Equation 13 gives the **natural frequency** in radians per second and in cycles per second: stiffer springs raise it, heavier masses lower it. A tall structure is many masses and many springs, and each of its modes has such a frequency; the lowest is the whole structure swaying, and a wind-turbine tower's is typically a fraction of a cycle per second.

Worked example 5: the two modes of the coffee and mug

For $A = \left[\begin{smallmatrix} -0.1 & 0.1 \\ 0.3 & -0.5 \end{smallmatrix}\right]$, the condition $A\,v = \lambda\,v$ has a non-zero solution only when the determinant of $A - \lambda\,I$ is zero: $(-0.1 - \lambda)(-0.5 - \lambda) - 0.1 \times 0.3 = 0$, which multiplies out to $\lambda^2 + 0.6\,\lambda + 0.02 = 0$. The quadratic formula gives $\lambda = \left(-0.6 \pm \sqrt{0.36 - 0.08}\right)/2$, so $\lambda_1 = -0.0354$ and $\lambda_2 = -0.5646$ per minute. Time constants $-1/\lambda$: 28.2 minutes and 1.77 minutes. For each, the shape comes from row one of $A\,v = \lambda\,v$: $-0.1\,v_1 + 0.1\,v_2 = \lambda\,v_1$, so $v_2 = (\lambda + 0.1)\,v_1/0.1$. Taking $v_1 = 1$: the slow mode is $v = \left[\begin{smallmatrix} 1 \\ 0.646 \end{smallmatrix}\right]$ and the fast mode $v = \left[\begin{smallmatrix} 1 \\ -4.646 \end{smallmatrix}\right]$. Check the slow one against row two: $0.3 \times 1 - 0.5 \times 0.646 = -0.023$, and $\lambda_1\,0.646 = -0.0229$; they agree.

Read the two shapes. The slow mode has both excesses positive with the mug at 65 per cent of the coffee's: the two cooling *together* to the room over half an hour. The fast mode has them opposite in sign: the coffee and mug *equalising* with each other in under two minutes. Every trajectory of this system is a mix of the two, and after a few minutes only the slow one is left – which is what the eye sees in a real mug of coffee, and what the Stiff Equations subsection meant by a fast part and a slow part.

Worked example 6: a natural frequency, and two masses

A 2 kilogram mass on a spring of 50 newtons per metre has $\omega = \sqrt{50/2} = 5$ radians per second, so $f_n = 5/(2\,\pi) = 0.80$ cycles per second and a period of 1.26 seconds. Now two equal masses between three equal springs, wall to mass to mass to wall. Each mass feels its own spring and the shared one: $m_s\,d^2x_1/dt^2 = -2\,k_s\,x_1 + k_s\,x_2$ and the mirror image for x_2. Try the two obvious shapes. In the **in-phase** shape $\left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]$ the masses move together, the shared spring never stretches, each mass feels one spring, and $\omega_1 = \sqrt{k_s/m_s} = 5$: substituting $x_2 = x_1$ into the first equation gives $-k_s\,x_1$ on the right, as claimed. In the **anti-phase** shape $\left[\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right]$ the masses move oppositely, the shared spring stretches twice as much, and substituting $x_2 = -x_1$ gives $-3\,k_s\,x_1$, so $\omega_2 = \sqrt{3\,k_s/m_s} = 8.66$ radians per second. Two masses, two modes, two natural frequencies; a tower with a million states has a million, and a reduced model keeps the first few.

Probability and Distributions

Probability: from counting to a number between zero and one

A pump has been logged every hour for a year. Of its 8,760 hours, it was off for 5,694, on for 2,628 and faulted for 438. Ask "what is the chance the pump is on at a randomly chosen hour" and the classical answer is a ratio: the number of hours that satisfy the condition over the number of hours there are, $2628/8760 = 0.3$. That ratio is a **probability**: a number between zero and one, where zero means never and one means always. An **event** is a condition on the outcome – "the pump is on", "the reading is above 70" – and the probability of an event is the fraction of outcomes in which it holds, when every outcome is equally likely, or the fraction of the time it has held, when the outcomes are observed rather than counted. Three rules follow from the ratio and are all the probability calculus this book uses:

- The probabilities of all the outcomes sum to one: $0.65 + 0.3 + 0.05 = 1$, because the pump is always in *some* state.
- Two events that cannot both happen – **mutually exclusive** – have probabilities that add: $P(\text{onorfaulted}) = 0.3 + 0.05 = 0.35$.
- The probability that an event does not happen is one minus the probability that it does: $P(\text{notfaulted}) = 1 - 0.05 = 0.95$.

Worked example 1: three probabilities from one log

From the log above, the probability that the pump is not off is $1 - 0.65 = 0.35$, which is the same as $P(\text{onorfaulted})$, as it must be: "not off" and "on or faulted" are the same event said two ways. The probability that it is off *or* on is $0.65 + 0.3 = 0.95$, which is the complement of faulted. And the probability that it is on *and* faulted is zero, because one pump is in one state at a time; that is what mutually exclusive means, and it is why the second rule could add.

Conditional probability: "given"

Often the question is not "how likely is this" but "how likely is this, *given* that". Of a fleet of 1,000 bearings at the same age, 50 are running hot and 950 are normal. A vibration alarm has fired on 45 of the hot ones and, falsely, on 95 of the normal ones. Given that a bearing's alarm has fired, how likely is it hot? Restrict the count to the bearings that satisfy the condition – the 140 with an alarm – and count the hot ones among them: $45/140 = 0.32$. That is a **conditional probability**, written $P(\text{hot} \mid \text{alarm})$ and read "the probability of hot given alarm", and the general form is the ratio of the joint count to the conditioning count:

$$P(A \mid B) = \frac{P(A \text{ and } B)}{P(B)} \tag{14}$$

In Equation 14, $P(A \text{and} B)$ is the probability that both events hold – $45/1000 = 0.045$ for hot and alarm – and $P(B)$ is the probability of the condition, $140/1000 = 0.14$. The division rescales the joint probability so that it is measured against the condition rather than against everything. The bar is read "given", and everything to its right is what you already know.

Bayes' rule: turning a conditional around

The same joint count can be conditioned the other way. $P(\text{alarm} \mid \text{hot})$ is the fraction of *hot* bearings whose alarm fired, $45/50 = 0.9$; that is a property of the sensor, and it is usually what a datasheet or a test bench gives you. What a twin needs is the reverse, $P(\text{hot} \mid \text{alarm})$ – what to believe about the bearing given what the sensor said. Writing the joint probability both ways, $P(A \text{and} B) = P(A \mid B)\,P(B)$ and $P(A \text{and} B) = P(B \mid A)\,P(A)$, and equating them gives the one relation every estimator in this book is built on:

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)} \tag{15}$$

Equation 15 is **Bayes' rule**, and the words attached to its symbols matter more than the symbols. With A the state and B the reading: $P(A)$ is what you believed about the state *before* the reading, the **prior**; $P(B \mid A)$ is how likely that reading would be *if* the state were A, the **likelihood**, a property of the sensor; $P(A \mid B)$ is what you believe *after* the reading, the **posterior**; and $P(B)$ is the total probability of the reading, found by adding $P(B \mid A)\,P(A)$ over every possible state, which is what makes the posteriors sum to one. The rule says: posterior is proportional to likelihood times prior. Every correction step in a state estimator is this line with the symbols dressed up.

```
prior      = {hot: 0.05, normal: 0.95}
likelihood = {hot: 0.9,  normal: 0.1}          # P(alarm | state)
unnorm     = {s: likelihood[s] * prior[s] for s in prior}
total      = sum(unnorm.values())                # P(alarm)
posterior  = {s: unnorm[s] / total for s in prior}
```

Worked example 2: Bayes' rule on the bearing, twice

Prior $P(\text{hot}) = 0.05$, $P(\text{normal}) = 0.95$, from the fleet counts. Likelihoods $P(\text{alarm} \mid \text{hot}) = 0.9$ and $P(\text{alarm} \mid \text{normal}) = 0.1$, from the sensor. The alarm fires. Top line for each state: hot, $0.9 \times 0.05 = 0.045$; normal, $0.1 \times 0.95 = 0.095$. Total $P(\text{alarm}) = 0.14$. Posterior $P(\text{hot} \mid \text{alarm}) = 0.045/0.14 = 0.32$ – the same 0.32 the direct count gave, which is the check that Bayes' rule is the counting argument in symbols. One alarm from a 90-per-cent sensor leaves the bearing more likely fine than hot, because hot bearings were rare to begin with; that is the prior doing its work, and it is why a twin diagnoses from a belief and not from a single reading.

A second alarm an hour later, taken as independent of the first given the bearing's state, starts from the new prior of 0.32: hot, $0.9 \times 0.32 = 0.29$; normal, $0.1 \times 0.68 = 0.068$; total 0.357; posterior $0.29/0.357 = 0.81$. Two alarms make it likely. The posterior of one step became the prior of the next, which is what the Markov section will call the belief.

Independence

Two events are **independent** when knowing one changes nothing about the other: $P(A \mid B) = P(A)$, or equivalently

$$P(A \text{ and } B) = P(A)\,P(B) \tag{16}$$

Equation 16 is the test as well as the definition: compute both sides from the counts, and the events are independent only if they agree. The two alarms above were treated as independent *given* the bearing's state, which is what let the second update start from the first's posterior; a state estimator assumes the noise in the model's step and the noise in the sensor are independent of each other and of everything earlier, and that assumption is the one most often violated in practice, by a sensor whose error drifts rather than resets.

Worked example 3: two sensors, dependent or not

Two pressure sensors on the same pipe each fail on any given day with probability 0.01. If they fail independently, the probability that both fail today is $0.01 \times 0.01 = 0.0001$, one day in ten thousand, which is why a second sensor is bought. Now suppose both are fed from one power supply that fails on one day in two hundred, 0.005, taking both sensors with it. The probability that both fail is now *at least* 0.005, fifty times the independent figure, and the test fails: $P(\text{both}) \geq 0.005$ but $P(A)\,P(B) = 0.0001$. The second sensor was bought for independence and the wiring took it away. Independence is a fact about the world to be checked, not a property to be assumed because there are two of something.

From events to quantities: the density

A thermometer in the coffee at minute three does not read 64.45. It reads 64.3, or 64.7, or 64.5, and if you read it a hundred times it reads a hundred slightly different numbers. The reading is a **random quantity**: one you cannot compute, only describe by how likely each value is. For a quantity that takes one of a few values the description is a list – the pump's $0.65, 0.3, 0.05$ – called a **probability mass function**. For a quantity that can take any value on a line, no single value has a probability of its own, and the description is a **density** $p(x)$: a non-negative function of the possible values whose *area* over an interval is the probability that the quantity lands in that interval.

$$P(\alpha \leq x \leq \beta) = \int_{\alpha}^{\beta} p(x)\,dx \tag{17}$$

Equation 17 asks for the area under a curve between two points, which is the integral of The Integration section. That is all a density is – a curve whose areas are probabilities – and the one condition on it is that the whole area is one, because the quantity has to land somewhere. The three rules of the first subsection carry over with sums replaced by integrals: areas over non-overlapping intervals add, and the area outside an interval is one minus the area inside.

Mean, variance, standard deviation. Two numbers summarise a distribution. The **mean** μ is where it balances, and the **variance** σ^2 is how far it spreads about that point, measured as the average squared distance. For a list they are weighted sums; for a density the sums become integrals:

$$\mu = \int x\,p(x)\,dx, \quad \sigma^2 = \int (x-\mu)^2\,p(x)\,dx \tag{18}$$

Both integrals in Equation 18 run over every value the quantity can take. The variance carries the quantity's units squared, which is awkward to think in, so its square root, the

standard deviation σ, is what gets quoted: a thermometer "good to half a degree" has $\sigma = 0.5$ degrees and a variance of 0.25 degrees squared. Every uncertainty figure in this book is a standard deviation unless it says variance. For a vector of quantities the variance becomes a table, the **covariance matrix** P: the two variances on the diagonal, and off it a number saying how the two move together, positive if one runs high when the other does, zero if knowing one says nothing about the other. It is symmetric, and every P, Q and R a state estimator carries is one.

```
mean = sum(x * p(x) for x in values)                    # a list
var  = sum((x - mean)**2 * p(x) for x in values)
sd   = sqrt(var)
# for a density, replace each sum by the integral over the line
```

Distributions: the two shapes this book uses

A **distribution** is a named family of densities, fixed by a few numbers. Two do all the work here.

The uniform distribution. Every value in an interval from a to b is equally likely and nothing outside it is possible. The density is flat, $p(x) = 1/(b-a)$ on the interval, so that its area is one, and zero elsewhere. Its mean is the midpoint, $(a+b)/2$, and its variance is $(b-a)^2/12$, so its standard deviation is about 0.29 of the width. It is the distribution of a reading you know only to within a rounding step, and of a parameter you know only to within a range.

The normal, or Gaussian, distribution. The bell curve, written $\mathcal{N}(\mu, \sigma^2)$:

$$p(x) = \frac{\exp\!\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}} \tag{19}$$

Equation 19 is fixed completely by its mean and variance: give those two numbers and the whole curve is known. Three areas under it are worth memorising: about 68 per cent of the probability lies within one standard deviation of the mean, 95 per cent within two, and 99.7 per cent within three. It is the default description of sensor noise, and for a reason rather than a habit: a disturbance made up of many small independent causes tends towards a bell curve whatever the causes' own shapes, and it is the only distribution under which a belief can be carried as a mean and a covariance alone, which is what makes a state estimator tractable. One more fact about it is used constantly: **a linear rule preserves it**. Multiply a Gaussian quantity by a constant a and add a constant b, and the result is Gaussian with mean $a\,\mu + b$ and variance $a^2\,\sigma^2$; the mean goes through the rule, the standard deviation is scaled by the multiplier, and the offset does not touch it. Where the Gaussian fails – a sensor that sticks, a quantity that cannot go negative, a process with rare large jumps – a state estimator built on it fails with it, and the book says so where it arises.

Worked example 4: a rounded reading as a uniform density

A sensor reports a reading rounded to the nearest ten units, so a displayed 15 means the true value is somewhere between 10 and 20 with no value more likely than another. The density is $p(x) = 0.1$ on the interval from 10 to 20, and its area is $0.1 \times 10 = 1$ as required.

The probability that the true value lies between 12 and 15 is the area of the rectangle over that span, $0.1 \times 3 = 0.3$. The mean is the midpoint, $\mu = 15$, which the first integral confirms: $\int_{10}^{20} 0.1\,x\,dx = 0.1 \times (400 - 100)/2 = 15$. The variance is the second integral, $\int_{10}^{20} (x - 15)^2 \times 0.1\,dx$; substitute $u = x - 15$, integrate u^2 from -5 to 5 to get 250/3, and multiply by 0.1 to get $\sigma^2 = 8.33$, which is $(20 - 10)^2/12$ as the family's formula says. The standard deviation is $\sigma = 2.89$ units, about a third of the interval's width.

Worked example 5: a thermometer as a Gaussian

The thermometer's reading of the coffee at minute three is Gaussian with mean 64.45 and $\sigma = 0.5$ degrees. The probability that a single reading lies within half a degree of the truth – between 63.95 and 64.95 – is the one-standard-deviation area, about 0.68; within one degree, about 0.95; within one and a half, 0.997. So roughly one reading in twenty is more than a degree off, and one in three hundred more than a degree and a half. Now push the reading through one Euler step of the cooling equation, $T_1 = (1 - h\,k)\,T_0 + h\,k\,T_\text{a}$, a linear rule with multiplier $a = 0.9$ and offset $b = 2$ at $h = 1$. By the linear rule fact, the predicted T_1 is Gaussian with mean $0.9 \times 64.45 + 2 = 60.0$ and standard deviation $0.9 \times 0.5 = 0.45$ degrees. The step *shrank* the uncertainty, because the cooling law pulls every trajectory toward the room and so pulls nearby trajectories together. Convert the reading to Fahrenheit instead, $F = 1.8\,T + 32$: the mean becomes 148.0 and the standard deviation $1.8 \times 0.5 = 0.9$; the offset of 32 moved the mean and left the spread alone.

State Machines with Probabilities: The Markov Chain

Boxes and arrows

Not every quantity a twin tracks is a number that slides. A pump is off, on, or faulted, and it jumps between those. You already know the picture for that: a **finite state machine** – boxes for the states, arrows for the transitions, exactly one box active at a time, and on each arrow the event that fires it. A watering controller is one; so is the lifecycle of a network connection. A finite state machine is a differential equation's discrete cousin: instead of "given where you are, here is how fast you move", it says "given where you are, here is where you go next".

Put a probability on every arrow

A twin usually cannot say *which* arrow fires next, only how likely each one is over the next interval. Write that likelihood on the arrow and the picture is a **Markov chain**. Figure 2 draws one for the pump, with a step of one hour.

Read Figure 2 from each state in turn. An idle pump stays idle with probability 0.7 and starts with 0.3. A running pump stops with 0.4, keeps running with 0.58, and fails with 0.02. A faulted pump is repaired and returned to idle with 0.1 and stays faulted with 0.9. The numbers on the arrows leaving any one state sum to one, because the pump will be in *some* state an hour from now.

The same picture as a table is the **transition matrix** Π: one row per present state, one column per next state, and Π_{ij} the probability of going from state i to state j in one step. Ordering the states Off, On, Faulted:

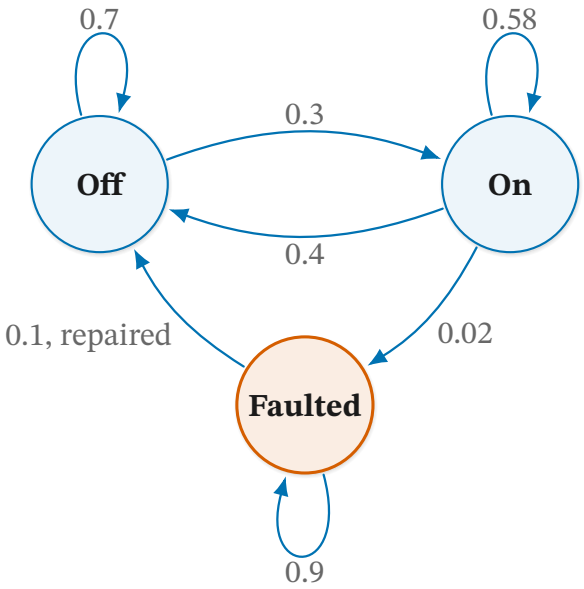


Figure 2: A three-state pump as a Markov chain. Each arrow carries the probability of taking it in the next hour, including the arrow from a state back to itself.

$$\Pi = \begin{bmatrix} 0.7 & 0.3 & 0 \\ 0.4 & 0.58 & 0.02 \\ 0.1 & 0 & 0.9 \end{bmatrix} \tag{20}$$

Each row of Equation 20 sums to one – a **row-stochastic** matrix – and that is the whole of what the chain knows. The table *is* the model; the picture is its documentation.

Worked example 1: from a state machine to its matrix

Start with a state machine you would write anyway: three states, and the transitions Off to On (a start command), On to Off (a stop), On to Faulted (a failure), Faulted to Off (a repair). Ask, for each state and each hour, how often each transition fires. Operating logs for this pump say it runs about thirty per cent of its idle hours, so $\Pi_{12} = 0.3$ and the rest of the row, $\Pi_{11} = 0.7$, is staying put. It stops in forty per cent of its running hours and fails in two per cent, so $\Pi_{21} = 0.4$, $\Pi_{23} = 0.02$, and staying running takes the rest, $\Pi_{22} = 0.58$. Repair takes about ten hours on average, so a faulted pump leaves that state with probability 0.1 per hour, to Off; $\Pi_{31} = 0.1$, $\Pi_{33} = 0.9$, and $\Pi_{32} = 0$ because a repaired pump is never returned straight into service. Three transitions that the state machine did not have – the self-loops – appeared because every row must sum to one; that is the one thing the probabilistic version adds to the picture you started from.

The belief, and prediction as one multiplication

If you knew the pump's state you would write it as a row with a one in that state's column and zeros elsewhere. If you do not know it, you write how likely each state is: a **belief**, b_k, a row of probabilities that sums to one. To predict the belief one step ahead, ask: how

much probability flows into each next state? Into state j flows, from every present state i, the probability of being in i times the probability of going from i to j. That sentence is the weighted-sum loop of the Many Quantities section with the row on the left:

$$b_{k+1} = b_k \, \Pi \tag{21}$$

One matrix multiplication per step, and Equation 21 is the discrete form of the Euler loop: present belief in, next belief out, the table in between. With a row on the left the two nested loops run column by column:

```
b = [1.0, 0.0, 0.0]                    # certain: Off
for each hour:
    b_next = [0.0, 0.0, 0.0]
    for j in 0 .. 2:                   # one entry per next state
        for i in 0 .. 2:               # probability flowing in from
↪   state i
            b_next[j] = b_next[j] + b[i] * Pi[i][j]
    b = b_next
```

Worked example 2: two prediction steps, and the leak

The pump was seen idle at hour zero: $b_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$. One step: $b_1 = b_0 \, \Pi$ picks out row one of the table, $\begin{bmatrix} 0.7 & 0.3 & 0 \end{bmatrix}$. The belief is no longer certain; the pump is probably still idle but may have started. Second step, column by column. Off: $0.7 \times 0.7 + 0.3 \times 0.4 + 0 \times 0.1 = 0.61$. On: $0.7 \times 0.3 + 0.3 \times 0.58 + 0 \times 0 = 0.384$. Faulted: $0.7 \times 0 + 0.3 \times 0.02 + 0 \times 0.9 = 0.006$. So $b_2 = \begin{bmatrix} 0.61 & 0.384 & 0.006 \end{bmatrix}$, which sums to one. The largest entry fell from 1 to 0.7 to 0.61 in two hours: **certainty leaks away with every step** you predict without looking. Run the multiplication on and the belief settles to a fixed row that no longer depends on where the pump started, at which point the twin knows only what the logs knew. This is a twin's staleness in arithmetic: a twin that has not heard from its pump for a while holds a belief, and the belief is worth less the longer the silence.

The Markov property, in words and in one line

Everything above assumed one thing: that the next state depends on the present state and the present input, and on nothing earlier.

$$P(x_{k+1} \mid x_k, u_k, x_{k-1}, u_{k-1}, \dots) = P(x_{k+1} \mid x_k, u_k) \tag{22}$$

Equation 22 is the **Markov property**. The history to the left of the bar may be as long as you like; only the last state and input on it matter. It is the reason a belief is a single row and not a log of every past reading, and the reason the loop of the earlier sections can hold one value of T rather than its whole history: the state is *defined* as whatever the future depends on. If the future depends on something the state omits – how long the pump has already been running, say – the property fails, and the repair is to add that quantity to the state, not to abandon the chain.

The transition density and the noisy state-space model

For a quantity that slides rather than jumps, the rows of the table become densities. The **transition density** $p(x_{k+1} \mid x_k, u_k)$ says how likely each next value is given the present value and input; it is one row of Π, with a curve in place of a list. Where does that curve come from? From the state-space model of the Many Quantities section, with noise added:

$$x_{k+1} = f(x_k, u_k, \theta) + w_k, \quad z_k = g(x_k) + v_k \tag{23}$$

Equation 23 is the discrete state-space model with two new terms. The first line is the step rule – $F x_k + G u_k$ for a linear system, or any other rule that gives the next state – with a **process noise** w_k added for everything the rule leaves out. The second is the output rule with a **measurement noise** v_k for everything the sensor gets wrong. For the coffee, $f(T_k) = T_k - h\,k\,(T_k - T_a)$, $g(T_k) = T_k$, and if w_k is Gaussian with standard deviation 0.3 degrees, the transition density is a bell curve centred on the Euler prediction with $\sigma = 0.3$. The whole apparatus of a state estimator – and the formal descriptions of a twin as a pair of coupled dynamical systems carried by a probabilistic graphical model [6] – is this one line, chosen so that the Markov property holds.

A chain seen through a noisy sensor

The twin does not see the pump's state. It sees a current sensor that reports "drawing current" or "no current", and the sensor is imperfect: it reports current with probability 0.95 when the pump is On, 0.05 when Off (leakage and noise), and 0.3 when Faulted (a stalled motor draws some). A Markov chain observed through a sensor like that is a **hidden Markov model**, and Figure 3 draws it.

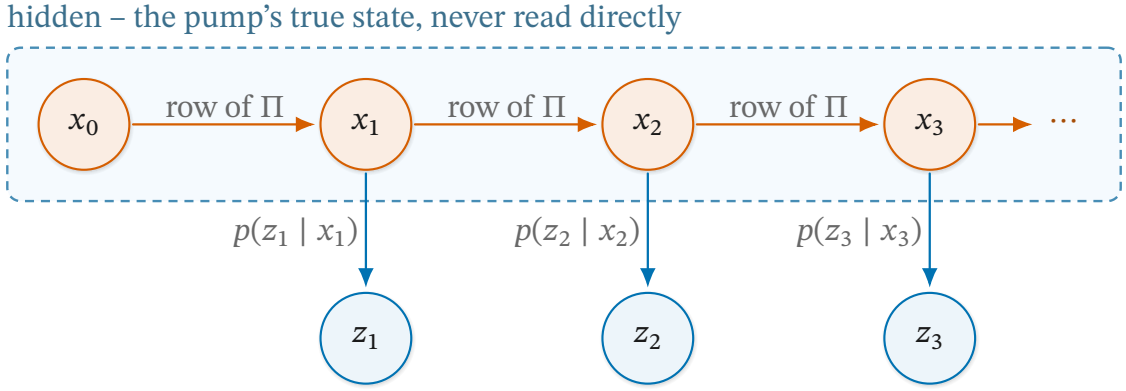


Figure 3: The pump chain seen through its current sensor. The states along the top are hidden; each reading below depends on its own state alone.

In Figure 3 every arrow along the top goes exactly one step – the Markov property drawn – and every reading hangs off its own state and nothing else. Nothing points back up: a reading reports on the pump, it never drives it. The twin's job is to maintain the belief over the hidden row using the observed one, and the recipe is the two steps already built: predict with Equation 21, then correct with Bayes' rule, using the sensor's probabilities as

the likelihood. A twin for a civil structure that carries its health state as a hidden variable and its sensor readings as the observed row is this figure with more states [7].

Worked example 3: one correction step

After two hours of silence the belief is $b_2 = \begin{bmatrix} 0.61 & 0.384 & 0.006 \end{bmatrix}$. The sensor now reports current. The likelihood of that reading in each state is $\begin{bmatrix} 0.05 & 0.95 & 0.3 \end{bmatrix}$. Bayes' rule, state by state, is prior times likelihood: Off, $0.61 \times 0.05 = 0.0305$; On, $0.384 \times 0.95 = 0.3648$; Faulted, $0.006 \times 0.3 = 0.0018$. The total is 0.3971, and dividing gives the posterior $\begin{bmatrix} 0.077 & 0.919 & 0.005 \end{bmatrix}$. One reading moved the belief from "probably idle" to "almost surely running", and the faulted entry is lower than before the reading, because a faulted pump draws current less often than a running one. Predict again with the table, correct again with the next reading, and you have a state estimator for a pump; do it with Gaussians and matrices instead of rows and lists, and you have a Kalman filter.

What a Solver and a Simulator Do

A solver runs the loop

A **simulator** is the program that runs a model forward in time. For a differential equation, the part of the simulator that does the running is the **solver**: it holds the state, chooses the step size, asks the equation for the rate, applies a step, and records the result. The loop of the From an Equation to a Loop section is a solver with the step chosen once, up front. A production solver chooses it as it goes, as the Better Solvers subsection described, and offers a choice of step formula; its interface is what you would design yourself – a function f that returns the rate given the state, an initial state, a time span, and a tolerance.

An event scheduler runs the discrete version

A discrete model – the pump chain, a conveyor, a message queue – has no rate to integrate. Nothing happens between events, so a solver that stepped through the quiet would waste every step. Its simulator is an **event scheduler**: a priority queue of pending events ordered by time, and a loop that pops the earliest, advances the clock to it, runs the handler, and pushes whatever new events the handler schedules.

```
queue = [(0.0, "start")]              # (time, event), earliest first
clock = 0.0
while queue not empty and clock < end:
    (t, event) = pop_earliest(queue)
    clock = t                         # time jumps; nothing happened
 ↪  between
    for (dt, next_event) in handle(event, state):
        push(queue, (clock + dt, next_event))
```

Time jumps from event to event, which is why a discrete simulation of a year of pump operation can finish in a millisecond, and why the same model in a fixed-step loop would take a step every second for a year. A discrete-event model is a model written for this loop; a conveyor or a print queue runs on it.

Worked example 1: the cost of a day of coffee

Cost of a run is **operations per step times number of steps**. One Euler step of the cooling rule is a subtraction, two multiplications and an addition: call it four operations. A day is 1440 minutes. At $h = 1$ minute that is 1440 steps and about 5,760 operations; at $h = 0.1$, 14,400 steps and 57,600 operations; at $h = 0.01$, 144,000 steps and 576,000 operations. A processor doing a billion simple operations a second finishes the finest of those in under a millisecond. For the coffee, step size is a question about accuracy alone; cost never enters. For the coupled coffee and mug, one step is a two-by-two times a vector – four multiplications and two additions – plus two additions for the step, so the cost roughly doubles and still does not matter. What changes the picture is dimension, in the third example.

Worked example 2: five events from the queue

Run the pump's scheduler for five events. The rules: a start command is due two hours after the pump last stopped; a running pump stops after one and a half hours; a failure is not scheduled here, to keep the trace short. Table 1 lists what the loop does.

Table 1: Five pops of the pump's event queue. The clock jumps to each event's time; nothing is computed between them.

Pop	Clock (h)	Event	State after	Pushed
1	0.0	start	Off	(2.0, switch_on)
2	2.0	switch_on	On	(3.5, switch_off)
3	3.5	switch_off	Off	(5.5, switch_on)
4	5.5	switch_on	On	(7.0, switch_off)
5	7.0	switch_off	Off	(9.0, switch_on)

Read Table 1 down the clock column: seven hours of pump operation in five iterations, each a pop, a handler and a push. The queue never holds more than one event here; a conveyor with twenty parts in flight holds twenty, and the priority queue is what keeps the earliest at the front. The cost of a discrete run is operations per event times number of events, and the number of events is a property of the model's activity, not of the clock.

The real-time factor

One number decides whether a model can live inside a twin. The **real-time factor** is simulated time covered divided by the wall-clock time it took to cover it:

$$RTF = \frac{t_{\text{sim}}}{t_{\text{wall}}} \tag{24}$$

Both times in Equation 24 are in the same unit, so the factor is a pure number. An RTF of one means the model keeps exact pace with the world. Below one, it falls behind: a twin running it can never catch up to the present and so cannot compare a prediction with the reading that has only now arrived. Above one, it runs ahead, and how far above one decides

what the twin can do with it: at $RTF = 10$ it can run ten what-if scenarios in the time the world takes to run one; at $RTF = 1000$ it can run a day of scenarios during the coffee break. The distinction between a shadow and a twin lives here: a model whose RTF is below one can shadow nothing, and any use beyond monitoring needs it well above one.

Worked example 3: which model can live inside the twin

The coffee model at $h = 1$ minute took about 5,760 operations for a day. At a billion operations a second that is $t_{\text{wall}} = 5.76 \times 10^{-6}$ seconds for $t_{\text{sim}} = 86{,}400$ seconds, so $RTF = 86{,}400/(5.76 \times 10^{-6}) = 1.5 \times 10^{10}$ – fifteen billion times faster than the world. It can live inside any twin, run every what-if anyone asks for, and never be noticed.

Now the million-state tower. One step is 10^{12} multiplications if the table were dense; a real finite-element solver exploits the zeros and does perhaps 10^9 operations a step, and needs a step of one millisecond to resolve the tower's highest mode of interest. A day is 8.64×10^7 steps and about 8.64×10^{16} operations. At 10^{10} operations a second on a good workstation that is $t_{\text{wall}} = 8.64 \times 10^6$ seconds, a hundred days, for one simulated day: $RTF = 0.01$. This model cannot live inside the twin, for any purpose. The answer is not a faster computer – a hundred-fold shortfall is not closed that way – but a different model: a reduced-order model that keeps the few modes the Many Quantities section described, with n in the tens, an RTF in the thousands, and an error that then has to be accounted for. The real-time factor is the number that forces that decision, and the reason a twin holds more than one kind of model at all.

Notation Used in This Book

The book reserves a small set of letters and uses them the same way in every chapter. Table 2 collects them, with the way each is read aloud, so that a symbol met in a chapter can be looked up here in one place.

Table 2: The letters the book reserves, how each is read aloud, and the section of this appendix that builds it.

Symbol	Read as	Meaning	Built in
T, $T(t)$	"T", "T at time t"	a quantity; the same quantity at a named moment	Quantities and Rates
dT/dt	"the rate of T"	how fast T is changing now; the derivative	Quantities and Rates
θ	"theta"	the parameters of a rule, spoken of together	Quantities and Rates
h	"the step"	the time one step of the loop advances	From an Equation to a Loop

Symbol	Read as	Meaning	Built in
x_k	“x at step k”	a quantity after k steps; a subscript always counts steps, never time	Quantities, Rates and Equations
x	“the state”	the record of everything the model must remember; dimension n	Many Quantities
u	“the input”	what is done to the system from outside: a command, a load	Many Quantities
z	“the output” or “the reading”	what a sensor reports	Many Quantities
w_k, v_k	“process noise”, “measurement noise”	what the rule leaves out; what the sensor gets wrong	The Markov Chain
f, g	“the step rule”, “the output rule”	next state from present state and input; reading from state	Many Quantities
A, A^T, A^{-1}, I	“A”, “A transpose”, “A inverse”, “the identity”	a table of coefficients; rows and columns swapped; the table that undoes it; the table that changes nothing	Many Quantities
B, C, F, G	“B”, “C”, “F”, “G”	the input table; the output table; the discrete step table; the discrete input table	Many Quantities
λ, v, ω	“lambda”, “v”, “omega”	a mode’s rate; its shape; a natural frequency in radians per second	Many Quantities
$p(x)$	“the density of x”	a curve whose areas are probabilities	Probability

Symbol	Read as	Meaning	Built in
μ, σ, σ^2	“mu”, “sigma”, “sigma squared”	mean; standard deviation, in the quantity’s units; variance	Probability
P, Q, R	“P”, “Q”, “R”	covariance of the state; of the process noise; of the measurement noise	Probability
$\mathcal{N}(\mu, \sigma^2)$	“normal with mean mu and variance sigma squared”	the Gaussian density	Probability
$p(x \mid z)$	“the density of x given z”	conditional density; the bar is “given”	Probability
Π, b_k	“the transition matrix”, “the belief at step k”	a row-stochastic table; a row of probabilities over the states	The Markov Chain
$\hat{x}$	“x hat”	an estimate of x; a hat always marks an estimate, never a true value	Probability
RTF	“the real-time factor”	simulated time covered per unit of wall-clock time	Solver and Simulator

Two of those letters clash with the symbols the book’s watering-rig case study uses, and the book keeps both rather than renaming either. Table 3 lists the rig’s symbols and says where each clash is resolved.

Table 3: The watering rig’s symbols beside the reserved letters they share, and how a reader tells them apart.

Rig symbol	Meaning for the watering rig	Clash, and how it is read
m, m_{t}	soil moisture; target moisture	none; the appendix uses m for a matrix’s row count only inside the Many Quantities section
V, $V_{\max}$	dose asked for; the largest dose	none

Rig symbol	Meaning for the watering rig	Clash, and how it is read
T_{r}, T_{c}	the sensor's reading period, seconds; the twin's check interval, hours	none; both are step sizes in this appendix's sense
k	the controller's gain, millilitres per point of deficit	the appendix's cooling constant is also k, local to the coffee example; in a subscript, k always counts steps. The context – a rig, a coffee, a subscript – says which
d, g	drying rate; dose response	g is also the book's output rule in the state-space model; the rig's g carries units of points per millilitre and the output rule carries none
r, $\hat{m}$	the residual; the twin's predicted moisture	none; $\hat{m}$ is the hat convention above, applied to the rig

The mapped-mathematics convention, restated as a table, is Table 4: what is typed in the book's source and what it becomes on the page.

Table 4: How the book's plain-text source spells mathematics, and what the rendered page shows for each spelling.

Typed in the source as	Becomes on the page	Example
an asterisk between two symbols	multiplication, usually shown as a thin space or a times sign	$a\,b$
a forward slash	division, as a fraction where the line is displayed	a/b
a caret	a power	a^2
an underscore	a subscript	x_k
a comma between two relations, on one line	two equations set side by side	$y_1 = T_1 - T_{\mathrm{a}}, y_2 = T_2 - T_{\mathrm{a}}$
semicolons between rows inside brackets	a matrix or a column vector	$\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$
the word `integral` with limits after it	an integral sign with those limits	$\int_{\alpha}^{\beta} p(x)\,dx$

Typed in the source as	Becomes on the page	Example
a hat, transpose or inverse spelled as a suffix	the hat, the superscript T, the superscript minus one	$\hat{x}$, A^{T}, A^{-1}

Three lines read aloud, as worked examples of the table. First, $x_{k+1} = f(x_k, u_k, \theta) + w_k$ is "x at step k plus one equals f of x at step k, u at step k and theta, plus the process noise at step k". Second, $P(\text{hot} \mid \text{alarm}) = 0.32$ is "the probability of hot given alarm is 0.32". Third, $b_{k+1} = b_k\,\Pi$ is "the belief at step k plus one equals the belief at step k times the transition matrix". A reader who can produce those three sentences from the symbols can read every equation in the book.

Where Each Section Is Used

Table 5 runs from each section of this appendix to the chapters and sections that lean on it. This is the one place the appendix names a chapter. Chapters not yet written are named by their role in the book and marked as planned.

Table 5: From each section of this appendix to the chapters and sections that use it. Planned chapters are named by role.

Appendix section	Leaned on by
Quantities, Rates and Equations	Chapter 2, Watering Strategies and Plant Digital Twin (the drying rate d and the dose response g as rate rules; the controller as a difference equation); Chapter 3, Model, Simulation and Simulator; the models chapter, the lumped first-principles model type (planned)
The Integration	Chapter 2, Plant Digital Twin (the predicted moisture between checks as an integral of the drying rate); the models chapter, where a closed form exists and where it does not (planned)
From an Equation to a Loop	Chapter 5, Sampling and Time (sampled time, the sampling interval as the step); the models chapter, the simulators half and its solver choices (planned); the state and estimator chapter, the prediction step (planned)

Appendix section	Leaned on by
Many Quantities at Once	Chapter 4, Digital Twin Configuration (the state as a record); Chapter 5, Sampling (a mode's natural frequency sets the interval); the models chapter, the high-fidelity and reduced-order model types and stiffness (planned); the state and estimator chapter, the state-space form throughout (planned)
Probability and Distributions	Chapter 4, Digital Twin Configuration (uncertainty as a field of state); Chapter 5, Sensors and Quality (accuracy as a standard deviation); the state and estimator chapter, the correction step and Bayes' rule (planned); the credibility chapter, stated uncertainty (planned)
State Machines with Probabilities	Chapter 4, Digital Twin Configuration and the `as_of` field (the belief and its leak); the models chapter, the discrete-state and probabilistic model types (planned); the state and estimator chapter, the hidden Markov model (planned)
What a Solver and a Simulator Do	Chapter 1, Six Value Patterns (what each pattern demands of the real-time factor); Chapter 3, Model, Simulation and Simulator; the models chapter, the simulators half and its real-time factor (planned); the standards chapter, co-simulation (planned)
Notation Used in This Book	every chapter from Chapter 2 onward

Read Table 5 down the right-hand column and one chapter appears against every row but the last: the state and estimator chapter is the one that needs all of this appendix, and a reader headed there should read it whole. Every other chapter needs one or two sections, which is why the appendix is arranged so that they can be read alone.

What this appendix established, against its objectives. A differential equation and an initial condition fix a trajectory, the rate at any state is one arithmetic step from the equation, and a difference equation is the same idea with the step built in (objective 1). The integral form says how much has accumulated, a closed form exists for the cooling law and few others, and Euler's step is one rectangle of that integral, with Heun's trapezium and the implicit step as the two improvements worth knowing (objective 2). A matrix times a vector is two loops, a transpose swaps rows and columns, a two-by-two inverts by a formula that fails when its determinant is zero, a system of equations is one matrix rule, the state-space form adds input and output tables, and a mode is a vector the matrix only stretches (objective 3). A probability is a ratio of counts, a conditional probability restricts

the count, Bayes' rule turns a conditional around, independence is a test, and a density's areas are probabilities, with the uniform and the Gaussian as the two shapes that matter (objective 4). A Markov chain is a state machine with a probability on every arrow, a belief predicts by one multiplication and corrects by Bayes' rule, and the Markov property is what lets a state be one row (objective 5). A solver runs the loop, a scheduler runs the queue, cost is operations per step times steps, and the real-time factor decides which model a twin can carry (objective 6).

Exercises

Each exercise names the learning objective it serves. Solutions follow, or a hint where the exercise is open.

1. *(Objective 1)* A vacuum flask has $k = 0.02$ per minute. Its tea is at 90 degrees in a room at 20. State the rate of change of the tea's temperature, name the state, the parameters and the initial condition, and say where the trajectory ends.
2. *(Objective 1)* A tank with a flat bottom of area 2 square metres is filled at 0.5 cubic metres a minute and drained at 0.3. Write the differential equation for the water level L and compute the rate. Which of the numbers are parameters, what is the state, and what is the difference equation for a level sensor read once a minute?
3. *(Objective 2)* Using the closed form, find when the paper cup ($k = 0.2$, $T_\mathrm{a} = 20$, $T(0) = 80$) reaches 30 degrees, and its time constant.
4. *(Objective 2)* Step the coffee ($k = 0.1$, $T_\mathrm{a} = 20$, $T_0 = 80$) forward two Euler steps with $h = 2$ minutes, then one Heun step with $h = 2$. Compare both with the closed-form 69.12 at minute two.
5. *(Objective 2)* A system has $k = 0.5$ per minute. What is the largest step at which Euler's loop stays stable? A probe in the same model has $k = 100$. What step does the pair need, what word describes the pair, and which method from the Better Solvers subsection removes the constraint?
6. *(Objective 3)* $A = \left[\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right]$ and $x = \left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]$. Compute $A\,x$, A^T and A^{-1}, and check the inverse by applying it to $A\,x$.
7. *(Objective 3)* Write the discrete state-space table $F = I + h\,A$ for the coffee and mug at $h = 0.5$ minutes, and apply it to $\left[\begin{smallmatrix} 60 \\ 0 \end{smallmatrix}\right]$. Why is the result not half of the $h = 1$ result?
8. *(Objective 3)* From the two modes of the coffee and mug, state the largest stable Euler step, and say which mode sets it.
9. *(Objective 4)* A reading is rounded to the nearest four units, so the true value is uniform on an interval of width four. Compute the probability it lies in the middle half of the interval, and its standard deviation.
10. *(Objective 4)* Repeat the bearing example with a better sensor: $P(\text{alarm} \mid \text{hot}) = 0.99$ and $P(\text{alarm} \mid \text{normal}) = 0.01$, prior 0.05. What is the posterior after one alarm, and why is it so much higher than 0.32?
11. *(Objective 4)* A thermometer has $\sigma = 0.5$ degrees. Out of 300 readings, about how many would you expect to be more than one degree off, and more than one and a half degrees off? A second thermometer of the same kind is added; state the assumption under which the probability that both are more than a degree off on the same reading is the product of the two, and one way the assumption could fail.
12. *(Objective 5)* From $b_2 = \left[\begin{smallmatrix} 0.61 & 0.384 & 0.006 \end{smallmatrix}\right]$, compute b_3 by one more multiplication with the pump's table, and check that it sums to one. Then, instead, suppose the sensor at

hour two reports *no* current; compute the posterior.

13. *(Objective 5)* The pump's failure probability is found to grow with the number of hours it has run since its last repair. The chain as drawn cannot express that. What fails, and what is the repair?
14. *(Objective 6)* A reduced-order tower model has $n = 20$, costs about 2,000 operations a step, and needs a step of 10 milliseconds. Compute its real-time factor on a machine doing 10^{10} operations a second, and say what a twin can do with it.
15. *(Objective 2, Objective 6)* Take the implicit Euler step for the coffee with $h = 10$ and with $h = 25$, compare each with the closed form and with the explicit step, and say what the implicit step buys a twin that must run a stiff model in real time.

Solutions and hints

Exercise 1 (solution). Rate $-0.02 \times (90 - 20) = -1.4$ degrees per minute. State T; parameters $k = 0.02$ and $T_a = 20$; initial condition $T(0) = 90$. The trajectory ends at the room temperature, 20, where the rate is zero.

Exercise 2 (solution). The level rises by the net inflow spread over the bottom: $dL/dt = (q_{in} - q_{out})/A_{tank}$, so $dL/dt = (0.5 - 0.3)/2 = 0.1$ metres per minute. The state is L; the three numbers are parameters – the area and the two flows – and the initial condition, which the exercise does not give, is whatever the level is when the clock starts. The difference equation is $L_{k+1} = L_k + 0.1$ with one step per minute, exact because the rate does not depend on L; the level is a straight line, like the drying soil.

Exercise 3 (solution). The excess must fall from 60 to 10, so $\exp(-0.2\,t) = 1/6$ and $t = \ln(6)/0.2 = 8.96$ minutes, half the ceramic cup's 17.9 because k doubled. The time constant is $1/k = 5$ minutes.

Exercise 4 (solution). Euler, step one: rate -6, so $T_1 = 80 - 12 = 68$. Step two: rate $-0.1 \times 48 = -4.8$, so $T_2 = 68 - 9.6 = 58.4$; that is minute four, so compare instead after one Euler step of $h = 2$ at minute two: 68 against 69.12, an error of 1.12. Heun with $h = 2$: start rate -6, guess 68, rate there -4.8, $T_1 = 80 + 1 \times (-6 - 4.8) = 69.2$, against 69.12, an error of 0.08 – fourteen times smaller for one extra rate evaluation.

Exercise 5 (hint). Stability needs $h < 2/k$, so four minutes for the system and about 0.02 minutes – a little over a second – for the probe. The pair needs the probe's step, everywhere, for the whole run, though the probe settles in seconds; that is stiffness. The implicit Euler step is stable at any step size, so the step can be chosen for the accuracy of the slow part alone.

Exercise 6 (solution). $A\,x = \left[\begin{smallmatrix} 1+2 \\ 3+4 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 3 \\ 7 \end{smallmatrix}\right]$. $A^{\mathsf{T}} = \left[\begin{smallmatrix} 1 & 3 \\ 2 & 4 \end{smallmatrix}\right]$. Determinant $1 \times 4 - 2 \times 3 = -2$, so $A^{-1} = \left[\begin{smallmatrix} -2 & 1 \\ 1.5 & -0.5 \end{smallmatrix}\right]$. Applying it to $\left[\begin{smallmatrix} 3 \\ 7 \end{smallmatrix}\right]$: row one $-2 \times 3 + 1 \times 7 = 1$, row two $1.5 \times 3 - 0.5 \times 7 = 1$. The original $\left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]$ comes back.

Exercise 7 (solution). $F = \left[\begin{smallmatrix} 0.95 & 0.05 \\ 0.15 & 0.75 \end{smallmatrix}\right]$. Applied to $\left[\begin{smallmatrix} 60 \\ 0 \end{smallmatrix}\right]$: $\left[\begin{smallmatrix} 57 \\ 9 \end{smallmatrix}\right]$. It is not half of $\left[\begin{smallmatrix} 54 \\ 18 \end{smallmatrix}\right]$'s *change* because the identity part of F does not scale with h: the change from $\left[\begin{smallmatrix} 60 \\ 0 \end{smallmatrix}\right]$ is $\left[\begin{smallmatrix} -3 \\ 9 \end{smallmatrix}\right]$, which *is* half of the $h = 1$ change $\left[\begin{smallmatrix} -6 \\ 18 \end{smallmatrix}\right]$. Halving the step halves each step's movement, not the state.

Exercise 8 (solution). The fast mode has $\lambda_2 = -0.5646$ per minute, so Euler is stable only for $h < 2/0.5646 = 3.54$ minutes. The slow mode, with $\lambda_1 = -0.0354$, would allow h up to 56 minutes on its own; the fast mode sets the step, which is the stiffness the section described.

Exercise 9 (solution). The density is 1/4 on the interval. The middle half has width two, so the probability is $2/4 = 0.5$. The variance of a uniform density of width W is $W^2/12$, here $16/12 = 1.33$, so $\sigma = 1.15$ units – again about 0.29 of the width.

Exercise 10 (solution). Hot: $0.99 \times 0.05 = 0.0495$. Normal: $0.01 \times 0.95 = 0.0095$. Total 0.059. Posterior $0.0495/0.059 = 0.84$. It is higher because the false-alarm rate fell from 10 per cent to 1 per cent: the normal bearings, which are nineteen times as common, now contribute nineteen times fewer false alarms, so the alarm carries more evidence. The prior did not change; the likelihood did.

Exercise 11 (solution). One degree is two standard deviations, so about 5 per cent of readings, fifteen of the 300, are more than a degree off; one and a half degrees is three standard deviations, so about 0.3 per cent, one reading. The product $0.05 \times 0.05 = 0.0025$ holds only if the two thermometers' errors are independent; a shared cold draught across both probes, or a shared reference voltage, makes both err the same way at once and the joint probability larger than the product.

Exercise 12 (solution). Off: $0.61 \times 0.7 + 0.384 \times 0.4 + 0.006 \times 0.1 = 0.5812$. On: $0.61 \times 0.3 + 0.384 \times 0.58 + 0 = 0.4057$. Faulted: $0.384 \times 0.02 + 0.006 \times 0.9 = 0.0131$. Sum 1.000. The largest entry is now 0.58, the leak continuing. For the reading of no current the likelihoods are one minus the current likelihoods, $\begin{bmatrix} 0.95 & 0.05 & 0.7 \end{bmatrix}$: Off 0.5795, On 0.0192, Faulted 0.0042, total 0.6029, posterior $\begin{bmatrix} 0.961 & 0.032 & 0.007 \end{bmatrix}$. The reading confirmed the belief rather than overturning it, and the faulted entry barely moved, because "no current" is consistent with both Off and Faulted.

Exercise 13 (hint). The Markov property fails: the next state depends on how long the pump has run, which is not in the state. The repair is to put it there – states such as "On, under 100 hours" and "On, over 100 hours", or a run-hours quantity beside the discrete state – and not to abandon the chain.

Exercise 14 (solution). A day is 8,640,000 steps of 10 milliseconds; at 2,000 operations each, 1.73×10^{10} operations, which is 1.7 seconds at 10^{10} a second. $RTF = 86{,}400/1.7 = 5 \times 10^4$. The twin can run fifty thousand simulated days in one real day, or a hundred what-if scenarios for the coming hour in seven seconds; prediction and decision support are open to it, at the cost of whatever accuracy the reduction gave up.

Exercise 15 (solution). With $h = 10$: $T_1 = (80 + 1 \times 20)/(1 + 1) = 50$, against a closed form of 42.07 and an explicit step of 20. With $h = 25$: $(80 + 2.5 \times 20)/3.5 = 37.1$, against 22.99 at minute 25 and an explicit -70. The implicit step is less accurate than the exact curve at these long steps but never unstable, so a stiff model can be stepped at the rate its slow part needs – and its real-time factor rises by the ratio of the two steps – instead of at the rate its fastest transient demands.

Further Reading

Grouped by section. The proofs this appendix omits are in the standard first-year texts of each subject, and any of them will do; the entries below are sources this book's own chapters draw on, and where in a twin each idea appears.

- **Equations, integration and the loop.** A foundations text for modelling cyber-physical systems derives Euler's error, introduces the Runge-Kutta family and mo-

tivates adaptive step size [1]. A coupled building-simulation system describes in practical terms what an adaptive-step solver does with a stiff mix of continuous and switching dynamics [2].

- **Vectors, systems and state space.** A twin of a laboratory incubator walks from a continuous state-space model through discretisation to the discrete system a state estimator runs on [3]; a physics-based twin of a floating wind turbine runs a Kalman filter over a linear structural model to estimate section loads along the tower [4]; and a structural-monitoring paper explains natural frequencies and mode shapes as the quantities operational modal analysis extracts from vibration data [5].
- **Probability and the Markov chain.** The formal picture of a twin and its asset as coupled dynamical systems carried by a probabilistic graphical model is set out for an unmanned aircraft [6], and applied to a civil structure with a dynamic Bayesian network carrying the observations-to-decisions flow [7]. Both are the hidden Markov model of this appendix at full scale.

References

[1] P. Carreira, V. Amaral, and H. Vangheluwe, Eds., *Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems*. Springer Nature, 2020. doi: 10.1007/978-3-030-43946-0.

[2] M. Wetter, K. Benne, H. Tummescheit, and C. Winther, "Spawn: Coupling Modelica Buildings Library and EnergyPlus to enable new energy system and control applications," *Journal of Building Performance Simulation*, vol. 17, no. 2, pp. 274–292, Mar. 2024, doi: 10.1080/19401493.2023.2266414.

[3] H. Feng, C. Gomes, and P. G. Larsen, "Model-Based Monitoring and State Estimation for Digital Twins: The Kalman Filter." arXiv, Apr. 2023. doi: 10.48550/arXiv.2305.00252.

[4] E. Branlard, J. Jonkman, C. Brown, and J. Zhang, "A digital twin solution for floating offshore wind turbines validated using a full-scale prototype," *Wind Energy Science*, vol. 9, no. 1, pp. 1–24, Jan. 2024, doi: 10.5194/wes-9-1-2024.

[5] F. Zonzini *et al.*, "Structural Health Monitoring and Prognostic of Industrial Plants and Civil Structures: A Sensor to Cloud Architecture," *IEEE Instrumentation & Measurement Magazine*, vol. 23, no. 9, pp. 21–27, Dec. 2020, doi: 10.1109/MIM.2020.9289069.

[6] M. G. Kapteyn, J. V. R. Pretorius, and K. E. Willcox, "A probabilistic graphical model foundation for enabling predictive digital twins at scale," *Nature Computational Science*, vol. 1, no. 5, pp. 337–347, May 2021, doi: 10.1038/s43588-021-00069-0.

[7] M. Torzoni, M. Tezzele, S. Mariani, A. Manzoni, and K. E. Willcox, "A digital twin framework for civil engineering structures," *Computer Methods in Applied Mechanics and Engineering*, vol. 418, p. 116584, Jan. 2024, doi: 10.1016/j.cma.2023.116584.