# The State Estimator

## Before You Start

**What you are assumed to know.** Three things, and only three. Matrices: how to multiply two small ones, transpose one, and invert a two-by-two. Ordinary differential equations: what $\frac{\mathrm{d}L}{\mathrm{d}t}$ means, and that an equation in it together with an initial condition fixes a trajectory. Basic integration: how to read an integral as an area, or as a weighted sum. A quantity indexed by a step number, so that $x_k$ and $x_0$ read as "the value at step $k$" and "the value at step zero", is notation rather than knowledge and is introduced where it is first used.

**What you are not assumed to know.** Any probability. The mean, the variance, the Gaussian, conditional probability and Bayes' rule are built from your integration in The Probability You Need, and nothing beyond that section is used. Any control theory. Any of the digital-twin research literature. Markov chains, which are defined here from scratch. Linear systems theory: observability, which this chapter needs twice, is defined here from scratch rather than recalled.

**What this chapter covers.** Why a digital twin that faithfully stores every sensor reading is still missing something; what that missing thing is; the small amount of probability the rest needs, built in one section; the one property of a well-chosen state, the Markov property, that makes it possible at all; the mathematics that supplies it, with the Kalman filter presented as a Markov chain observed through a noisy sensor; and five real systems built this way. The spine is one running example, a rooftop water tank, carried from a three-state chain through a fully worked filter to a nonlinear version posed as a problem. The control-engineering vocabulary for all of it – plant, controller, observer – is collected in one section near the end rather than used throughout.

**What this chapter deliberately does not cover.** Smoothing, which revisits an earlier estimate once later data has arrived; everything here is *filtering*. System identification, where the model comes from. The derivation of the Kalman gain, stated and interpreted rather than proved. Learned estimators, deferred to the closing section.

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

1. **Write** a discrete-time state-space model for a small physical system, naming its state, input, process noise and measurement noise.
2. **State** the Markov property in words and as an equation, **check** whether a proposed state satisfies it, and **explain** why it is what lets an estimator discard every past sample and still be right.
3. **Execute** a scalar Kalman filter by hand for several steps, reading each step as an operation on a Markov chain, and **predict** which way the gain will move before you compute it.
4. **Augment** a state vector to estimate a quantity no sensor measures, and **explain** which entry of the covariance matrix makes that possible.
5. **Choose** between a Kalman filter, an extended or unscented Kalman filter, a particle filter and a moving-horizon estimator for a stated system, and **defend** the choice.
6. **Diagnose** a misbehaving estimator from its innovation sequence, distinguishing a tuning fault from an unobservable state from a genuine anomaly in the physical system.

## 1. Why a Twin Needs a State Estimator

Picture a digital twin of a rooftop water tank serving an apartment building. A pump fills it, residents draw from it, and an ultrasonic sensor in the lid reports the water level once a minute over a wireless link. The twin writes every reading to a time-series database, a dashboard plots it, and an alarm fires below thirty centimetres.

This is a reasonable piece of software, and it is not yet a state estimator. Three gaps break it.

**The first gap is noise.** The sensor is accurate to about two centimetres, which sounds excellent until the alarm threshold is crossed by noise alone and the building manager is woken at three in the morning by a tank that is fine. A rolling average helps, and buys the improvement with lag: a ten-minute average is a statement about the level ten minutes ago, the wrong answer to give a pump controller. The twin needs a number that is both quiet and current, and averaging trades one for the other.

**The second gap is the quantities nobody measures.** The building manager does not much care about the level. She wants the *draw* – how fast residents are using water – because that tells her whether a pipe has started leaking and whether the pump schedule still fits the building. There is no draw meter, and fitting one means a plumber, a shutdown and an invoice. Yet the draw is present in the data: the level falls, and how fast it falls is the draw. The twinning literature names this split, calling a quantity *measurable* when a sensor reports it directly and *unmeasurable* when it matters and no instrument reads it [1].

**The third gap is the samples that never arrive.** The link drops, the sensor's battery browns out in cold weather, a firmware update takes it offline for six minutes. Meanwhile the pump runs, and the twin's picture of the tank is frozen at a reading growing staler by the second. A twin whose only source of truth is the sensor feed has nothing to say during an outage, which is often when somebody needs it to speak.

All three gaps have one shape. In each, the twin holds a second source of information it is not using: **it knows how the tank behaves.** A model can say what the level *ought* to be a minute from now, it can keep saying so through an outage, and the gap between what it says and what the sensor reports carries information about what neither measures alone.

A **state estimator** combines those two sources. It holds a running best guess of the system's internal state, pushes that guess forward through a model of the system's behaviour, and nudges it towards each measurement as it arrives, weighting model against sensor by how much each deserves to be trusted. It is a *fusion* of a physical model with an instrument, not a filter in the signal-processing sense of removing high frequencies. That it can hold a *running* guess at all, rather than the whole history of readings, rests on a property of the state that The State-Space Picture names and the rest of the chapter leans on: the Markov property.

This is not an embellishment on the digital-twin idea but the mechanism underneath it. Work that builds twins rather than frameworks reports that it is the ability to *detect* the physical system's states that enables everything a twin is bought for, so estimating them accurately through noise, network delay and missing signals is the basis on which the rest of the twin stands [2]. Control engineers have had a vocabulary for this arrangement for decades; the chapter keeps it out of the way until The Plant and the Controller, near the end, so that the estimator can be learned first on its own terms.

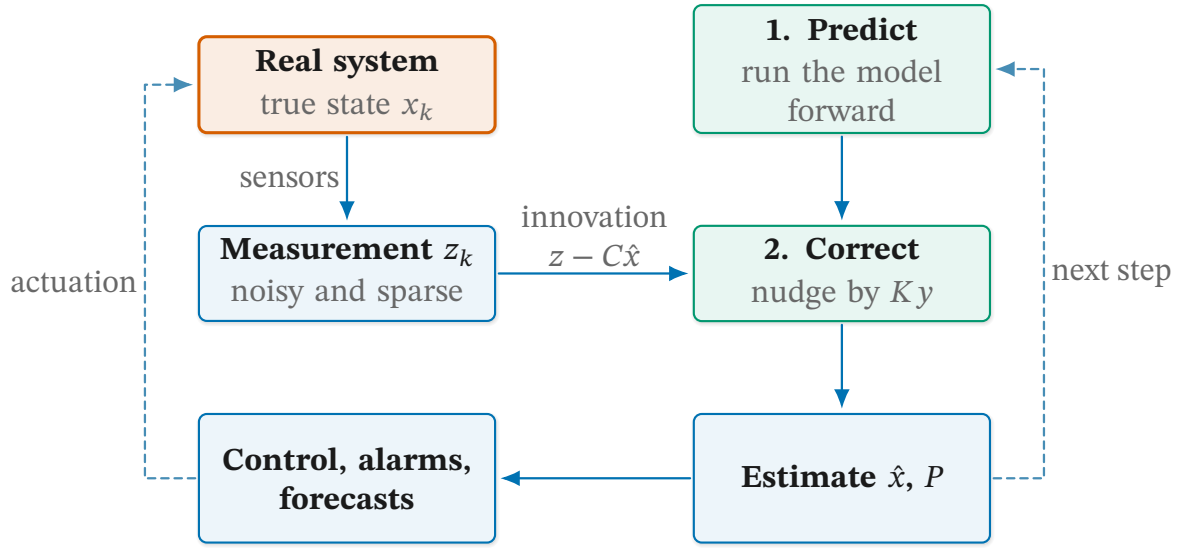



Figure 1: The estimator between the physical system and everything the twin does with it.

Figure 1 shows where it sits, with its corrected output serving as both the twin's answer and the starting point of the next prediction. The rest of this chapter fills in the two boxes at its centre.

## 2. The Probability You Need

This chapter uses probability and assumes none. Everything it needs fits in this section, and all of it is built from the integration you already know. Read it once now; the later sections point back to it by name.

**A random quantity and its density.** The tank's level next minute is not a number you can compute. It is a number you can only describe by how likely each value is, and the tool for that is a **probability density** $p(x)$: a non-negative function whose integral over an interval is the probability that the quantity falls in that interval.

$$\Pr(\alpha \le x \le \beta) = \int_{\alpha}^{\beta} p(x)\,\mathrm{d}x$$

The integral of $p(x)$ over the whole line is one, because the quantity has to be somewhere. Nothing more is asked of a density than that. When the quantity can take only a few values, the integral becomes a sum and the density becomes a short list of probabilities; the next section uses that version first.

**Mean and variance.** Two integrals summarise a density. The **mean** $\mu$ is where the density balances, and the **variance** $\sigma^2$ is how spread out it is about the mean.

$$\mu = \int x\,p(x)\,\mathrm{d}x, \qquad \sigma^2 = \int (x-\mu)^2\,p(x)\,\mathrm{d}x$$

The square root of the variance, the **standard deviation** $\sigma$, has the same units as the quantity, so a sensor accurate to two centimetres has $\sigma = 2$ and a variance of four. For a

vector of quantities the variance becomes a **covariance** matrix $P$: variances on the diagonal, and off the diagonal a number saying how much two components move together. A zero off-diagonal entry means knowing one component tells you nothing about the other; a negative one means that when one is higher than expected the other tends to be lower. Every $Q$, $R$, $P$ and $S$ in this chapter is a covariance.

**The Gaussian.** One density does almost all the work in this chapter.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

This is the **Gaussian**, or bell curve, written $\mathcal{N}(\mu,\ \sigma^2)$. It is completely fixed by its mean and its variance: give those two numbers and the whole curve is known. Two facts about it are used over and over. First, a linear rule preserves it: multiply a Gaussian quantity by a constant and add another constant, and the result is Gaussian, with the mean carried through the same rule and the variance scaled by the square of the multiplier. Second, the product of two Gaussian densities in the same variable is, up to a constant factor, another Gaussian, and it is narrower than either. The Kalman filter is these two facts applied once each per minute. Why the Gaussian is the default assumption is a fair question: noise made of many small independent causes tends towards one, and it is the only assumption under which a belief can be carried as a mean and a covariance alone. The chapter names where it fails.

**Conditional probability and Bayes' rule.** Two quantities $x$ and $z$, say a state and a reading, have a joint density $p(x, z)$ over pairs. The **conditional density** $p(x \mid z)$ is the density of $x$ once $z$ is known.

$$p(x \mid z) = \frac{p(x, z)}{p(z)}$$

Read the bar as "given". The same joint density can be cut the other way, $p(x, z) = p(z \mid x)\, p(x)$, and substituting gives the one relation every estimator in this chapter is built on.

$$p(x \mid z) = \frac{p(z \mid x)\, p(x)}{p(z)} \tag{1}$$

Equation 1 is **Bayes' rule**. On the right, $p(x)$ is what you believed about the state before the reading, the **prior**; $p(z \mid x)$ is how likely that reading would be if the state were $x$, the **likelihood**; and $p(z)$ is a constant that makes the left-hand side integrate to one. On the left, $p(x \mid z)$ is what you believe after the reading, the **posterior**. Every correction step in this chapter is this line with the symbols dressed up.

**Independence.** Two quantities are **independent** when knowing one changes nothing about the other, so that $p(x, z) = p(x)\, p(z)$. The chapter assumes the process noise and the measurement noise are independent of each other and of everything that came earlier. That is what lets the noise at one step be forgotten at the next, and it is the probabilistic small print behind the Markov property the next section states.

## 3. The State-Space Picture

Before an estimator can estimate anything, the system has to be written in a particular shape: the **discrete-time state-space model**. The shape has two halves, a rule for how the state moves and a rule for what a sensor reads, and this section builds each one twice – first as a table of probabilities, which is a version a software engineer already knows under another name, then as the matrix equation the rest of the chapter uses.

**The state, and the property that makes it one.** The **state** is the smallest set of numbers that, together with all future inputs, determines all future behaviour. "Smallest" is doing real work there. The tank's state is its water level, one number. Its colour and the weather are not state, because knowing them tells you nothing more about what the level will do next. A sensor reading is not state either. It is a *view* of the state, taken through an imperfect instrument.

Hidden in that definition is a claim about *history*. If the present state determines the future, then how the system arrived at its present state does not matter: two tanks that both hold 120 centimetres right now will do the same thing next minute whether one was filled an hour ago and the other five minutes ago. Once noise is admitted the future is no longer determined, but the claim survives in probabilistic form. The *probability distribution* of the next state depends on the present state and the present input, and on nothing earlier.

$$p(x_{k+1} \mid x_0, \dots, x_k,\ u_0, \dots, u_k) = p(x_{k+1} \mid x_k, u_k) \tag{2}$$

Equation 2 is the **Markov property**, and a sequence of states that has it is a **Markov chain**. The left-hand side conditions on the entire past; the right-hand side conditions on the present alone; the property says they are equal. Read it as a promise the state makes to the estimator: *everything the past has to say about the future, I already carry.* A state that keeps that promise can be estimated recursively, one step at a time, with the past thrown away. A state that breaks it cannot, and the standard repair is to enlarge the state until the promise holds again, which is what the Second Worked Example does.

**A Markov chain is a state machine with probabilities on the arrows.** Take a coarse version of the tank whose state is one of three words, *low*, *normal* or *high*, watched over a minute with its pump controller running. A state machine would draw three boxes and arrows between them. A Markov chain draws the same boxes and puts a probability on each arrow: from *normal*, say, the tank stays *normal* with probability 0.90, falls to *low* with probability 0.05 and rises to *high* with probability 0.05.

$$T = \begin{bmatrix} 0.90 & 0.10 & 0.00 \\ 0.05 & 0.90 & 0.05 \\ 0.00 & 0.10 & 0.90 \end{bmatrix}$$

Rows are the present state, in the order *low*, *normal*, *high*; columns are the next state; each row sums to one because the tank has to end up somewhere. This is a **transition matrix**, and the state machine it describes is a Markov chain. Where the machine would hold one current state, the chain holds a **belief**: a row of three probabilities $\pi_k$ saying how likely each state is at step $k$. Prediction is then one matrix multiplication, $\pi_{k+1} = \pi_k\, T$. Start certain that the tank is *normal*, $\pi_0 = [\,0,\ 1,\ 0\,]$, and after one step the belief is $[\,0.05,\ 0.90,\ 0.05\,]$;

after two it is $[\,0.09,\ 0.82,\ 0.09\,]$. Certainty leaks away a little each step, and nothing but a measurement can put it back. Hold on to that picture: it is the prediction step of every estimator in this chapter, and the Kalman filter's covariance growing by $Q$ each minute is the same fact in continuous clothing.

**From a table to an equation.** The tank's real level is not one of three words but a real number, so a table with a row per state is out of the question. The transition matrix becomes a **transition density**, $p(x_{k+1} \mid x_k, u_k)$: for each present state and input, a probability distribution over where the state lands next. Writing that density down directly is awkward. Writing a rule that *produces* it takes one line, and that line is the state-space model. Two rules, then, at each step $k$. One says how the state moves; one says what a sensor would read.

$$\begin{aligned} x_{k+1} &= A\,x_k + B\,u_k + w_k \\ z_k &= C\,x_k + v_k \end{aligned} \tag{3}$$

The first line of Equation 3 is the **process model**: $A$ says how the state evolves on its own, $B$ says how a known input $u_k$ pushes it, and $w_k$ is the **process noise**, the honest admission that the model leaves things out. The second line is the **measurement model**: $C$ says which parts of the state a sensor sees and in what units, and $v_k$ is the **measurement noise**. The state has $n$ components and the measurement has $p$, and the interesting case throughout is $p$ smaller than $n$ – fewer sensors than unknowns. Both noise terms are taken as zero-mean and independent, with covariance $Q$ for the process and $R$ for the measurement.

The connection to the transition density is direct. Fix $x_k$ and $u_k$, and the only random thing left on the first line is $w_k$, so $x_{k+1}$ is a fixed point $A\,x_k + B\,u_k$ plus a zero-mean disturbance of covariance $Q$. The second line, read the same way, says a reading is a fixed point $C\,x_k$ plus a disturbance of covariance $R$. Under the Gaussian assumption the chapter adopts in The Kalman Filter, both are bell curves.

$$p(x_{k+1} \mid x_k, u_k) = \mathcal{N}(A\,x_k + B\,u_k,\ Q) \tag{4}$$

$$p(z_k \mid x_k) = \mathcal{N}(C\,x_k,\ R)$$

Equation 4 is the first line of Equation 3 read as a Markov chain: the transition density, with $Q$ the spread of each arrow. The line beneath it is the second, the **measurement likelihood** $p(z_k \mid x_k)$: how probable a given reading is if the state has a given value. It depends on the present state alone, which is the measurement's own version of the Markov property.

**A chain you cannot see is a hidden Markov model.** The state is never read directly, only through the second line. A Markov chain observed through a noisy measurement of this kind is a **hidden Markov model**, and the twinning literature draws the discrete-time state-space model in exactly that form, rewriting the state transition equation as a transition probability density and the measurement equation as the conditional density of the measurement given the state [3]. Figure 2 shows the picture: the state runs along the top as a chain with one arrow per step, and each measurement hangs off its own state and nothing else.

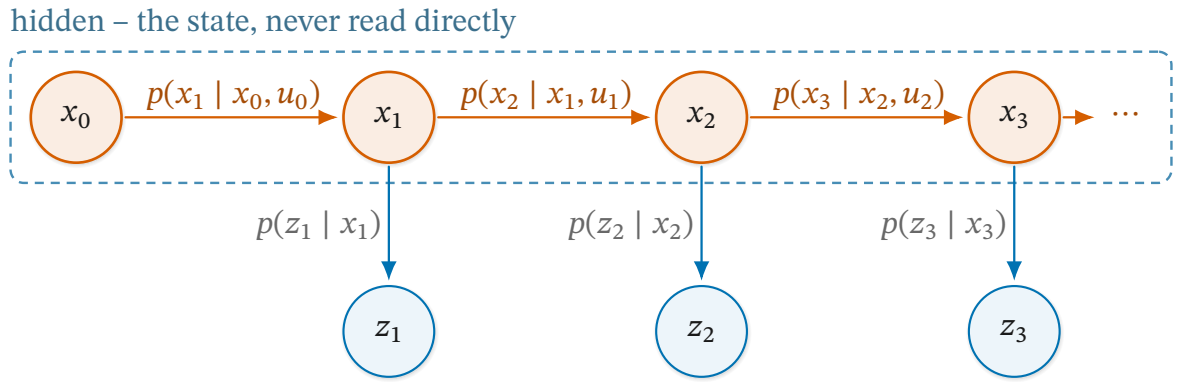



Figure 2: The state-space model drawn as a hidden Markov chain, with states along the top and each measurement depending on its own state alone.

Two features of the picture carry the rest of the chapter. There is no arrow from a measurement back up to a state: measurements report on the system, they never drive it. And no arrow along the top skips a state: the chain has no memory beyond one step, which is Equation 2 drawn rather than written. Estimation is the problem of reading the top row given only the bottom one.

**The tank as a chain.** Start where you would start with any physical system: a differential equation. The level rises at the pump's rate while the pump is on and falls at the draw's rate always, so with $b$ the centimetres per minute the pump adds, $d$ the draw in centimetres per minute, and $u(t)$ equal to 1 when the pump is commanded on and 0 when off,

$$\frac{\mathrm{d}L}{\mathrm{d}t} = b\,u(t) - d \tag{5}$$

The twinning literature introduces its physical models in exactly this form and describes the state as the model's *memory*, what you need to know about the condition the system has reached in order to continue it, which for a differential equation is its initial condition [4]. A sensor that reports once a minute cannot see Equation 5 between readings, so take one step of Euler's method with a step of one minute: the level at the next reading is the level now plus one minute's worth of the right-hand side. That is the discretisation the calibration literature writes down before it can compare a model with sampled data [1]. Write $L_k$ for the level in centimetres at step $k$ and $u_k$ for the pump command at that step, add a noise term for everything the equation leaves out, and let the sensor read the level directly.

$$\begin{aligned} L_{k+1} &= L_k + b\,u_k - d + w_k \\ z_k &= L_k + v_k \end{aligned} \tag{6}$$

Comparing Equation 6 with the general form, the tank has $A$ equal to one, $B$ equal to $b$, and $C$ equal to one, all scalars because the state is a single number. Read as a chain, its transition density is a bell curve centred on $L_k + b\,u_k - d$ with variance $Q$: the level next minute is where the pump and the draw would put it, blurred by the model's own error. Treating the draw $d$ as a known constant is a lie, and it is a lie that breaks the Markov property. If the draw is really drifting, then the level's recent history says something about where the draw

has drifted to, and so about the future, that the present level alone does not. The Second Worked Example is about stopping telling it.

## 4. Estimation as Predict and Correct

**Batch** estimation waits until all the data is in and finds the trajectory that best explains the whole record. It is the more accurate of the two approaches and useless for a running twin, which has to answer now.

**Recursive** estimation keeps one current estimate and updates it as each sample arrives, discarding the sample afterwards, so the whole past is compressed into the estimate and its uncertainty. In the online case the samples are not available as a batch, so the algorithm must work from the model and whatever samples exist at the current step [5]. Its cost per step is fixed however long the twin has been running, which is what makes it deployable.

That compression is not a trick of implementation. It is Equation 2 at work: because the present state carries everything the past has to say about the future, a belief about the present state carries everything the past *samples* had to say about it. Choose a state that breaks the property and no recursive estimator will be right, however carefully it is coded.

Every recursive estimator here has the same two-part skeleton, and both parts are operations on the hidden Markov chain of the previous section.

**Predict.** Take the belief about the present state and push it through the transition density. Where the three-state chain did this with one matrix multiplication, the continuous version averages the one-step rule over every place the state might currently be.

$$p(x_{k+1} \mid z_{1:k}) = \int p(x_{k+1} \mid x_k, u_k)\, p(x_k \mid z_{1:k})\, \mathrm{d}x_k \tag{7}$$

Equation 7 is the **Chapman-Kolmogorov equation**, and the tutorial treatment of the Kalman filter for digital twins derives the filter's prediction step from it directly [2]. $z_{1:k}$ is shorthand for every measurement up to step $k$, so the left-hand side is the belief about the *next* state given everything seen so far. Read the integral the way you read any other: for each place $x_k$ the state might now be, multiply the probability of being there, $p(x_k \mid z_{1:k})$, by the probability of stepping from there to $x_{k+1}$, and add up over all places. The estimate moves, and the uncertainty grows, because a step of the model's imperfection – the spread of the transition density – has been folded in.

**Correct.** A measurement arrives. Bayes' rule multiplies the predicted belief by the likelihood of what was actually read.

$$p(x_{k+1} \mid z_{1:k+1}) = \eta\, p(z_{k+1} \mid x_{k+1})\, p(x_{k+1} \mid z_{1:k}) \tag{8}$$

The constant $\eta$ in Equation 8 only makes the result sum to one; the work is done by the two factors after it, which raise the probability of states that would have produced the reading and lower the probability of states that would not. In practice: compute what the measurement *should have been* given the prediction. The difference is the **innovation**, and it is the only genuinely new information in the sample. Move the estimate some fraction of the way towards the measurement, and shrink the uncertainty.

That structure recurs wherever a filter is described as a fusion device: a prediction step that estimates the state and an update step that brings the sensors in, each weighted by its reliability [6]. The naive version of such weighting averages several sensors by accuracy; the estimator's version weights a *model prediction* against the sensors, which is what fills gaps a sensor average cannot. The formal account of twins arrives at the same two terms from its graphical model and calls the result a predictor-corrector update policy of the kind seen in hidden Markov models, Kalman filtering and particle filtering alike [7]. In general neither Equation 7 nor Equation 8 has a closed form, and *recursive Bayesian filtering* is the name for approximating them by alternating a transition step with an update step [3]. The rest of the mathematics in this chapter is a catalogue of ways to do that approximation, and the first is the one case where none is needed.

The whole design problem is the word *fraction*. Move all the way to the measurement and you have thrown the model away and kept the noise. Move not at all and you have thrown the sensor away and will drift.

## 5. The Kalman Filter

Suppose the transition density and the measurement likelihood are both Gaussian and both linear, exactly as Equation 4 wrote them, with $Q$ and $R$ known. Then something convenient happens to the two chain equations of the previous section. The two Gaussian facts from The Probability You Need do the work: a linear rule applied to a Gaussian gives a Gaussian, and the product of two Gaussians is a Gaussian, so the belief stays Gaussian for ever and is fully described by two objects: a mean and a covariance. The integral in Equation 7 and the product in Equation 8 both collapse to matrix arithmetic on those two objects. That arithmetic is the Kalman filter, and under those two assumptions its choice of fraction is provably best. "Best" means smallest error variance among all estimators, not merely among linear ones.

Predict:

$$\hat{x}_k^- = A\,\hat{x}_{k-1} + B\,u_k$$
$$P_k^- = A\,P_{k-1}A^\top + Q$$

The first line applies the process model to the estimate; the second applies it to the uncertainty and adds $Q$. This is Equation 7 for a Gaussian belief: the mean moves along the chain's arrow, and the covariance is the old covariance carried through the transition plus the transition's own spread. So $P^-$ is always at least as large as $P$ was: prediction always loses certainty, exactly as the three-state belief leaked certainty with every multiplication by $T$. An estimator left predicting with no measurements has its uncertainty grow without bound, which is the formal version of "the twin is guessing now".

Correct:

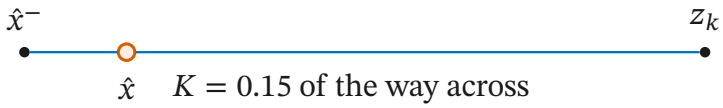
$$y_k = z_k - C\,\hat{x}_k^-$$
$$S_k = C\,P_k^-\,C^\top + R$$

$$K_k = P_k^-\,C^\top\,S_k^{-1} \tag{9}$$

$$\begin{aligned} \hat{x}_k &= \hat{x}_k^- + K_k\, y_k \\ P_k &= (I - K_k C)\, P_k^- \end{aligned} \tag{10}$$

Four relations, and the middle one carries the idea. Read the gain in Equation 9 as a ratio: the numerator is how uncertain the *prediction* is, the denominator $S$ is how uncertain prediction and *sensor* are together. The gain is the prediction's share of the total doubt. Shaky model and good sensor gives $K$ near one and the estimate jumps almost to the measurement; trusted model and noisy sensor gives $K$ near zero and the measurement barely moves anything. The two lines of Equation 10 then spend that gain twice over, moving the estimate and shrinking the uncertainty by the same factor. They are Equation 8 for Gaussians: multiplying the predicted bell curve by the likelihood bell curve gives a narrower bell curve whose centre has shifted towards the measurement by exactly $K$ times the innovation, and the denominator that made Bayes' rule sum to one has disappeared into the algebra. In the scalar case this collapses to arithmetic you can hold in your head, with $K = \dfrac{P^-}{P^- + R}$, $\hat{x} = \hat{x}^- + K\,y$ and $P = (1 - K)\,P^-$. Here $0 \le K \le 1$ always, so the estimate lands somewhere on the segment between prediction and measurement.

**(a)** small gain – the sensor is the noisier of the two




**(b)** large gain – the prediction is the less certain of the two




Figure 3: The gain read as a position on the segment from prediction to measurement.

Figure 3 draws that segment twice. In both panels the estimate lies between prediction and measurement, and the gain is the fraction of the distance travelled.

Two consequences explain what you will see in the next section.

- **The gain is not a constant you tune.** It is recomputed every step from the current uncertainty. A filter that has started recently, or has come back from an outage, has a large $P^-$ and grabs at measurements; one running well has a small $P^-$ and ignores most of the noise. The covariance recursion contains no $z_k$, so the whole sequence of gains could be computed before the system is switched on: the estimate depends on the data, the confidence does not.
- **The two limits are what you already knew.** A perfect sensor, $R \to 0$, drives the gain to one and the filter becomes the sensor; a perfect model, $Q \to 0$, drives it to zero and the filter becomes the model.

## 6. Worked Example: Three Minutes of a Rooftop Tank

**The setup.** One step is one minute. The pump adds $b = 4.0$ centimetres per minute when on. The draw is taken as constant at $d = 1.5$. The sensor has a standard deviation of two centimetres, so $R = 4.0$, because $R$ is a *variance*. The model is good but not perfect, and we allow it half a centimetre of slop per minute, so $Q = 0.25$.

**The starting point.** Somebody read 120 centimetres off a dipstick, by eye, in poor light: $\hat{L}(0) = 120.0$ with $P(0) = 9.0$, a standard deviation of three centimetres. Starting a filter means supplying both numbers, and the second matters as much as the first.

**Minute 1, pump off**, so $u_k = 0$. Predict: the chain's one-step rule carries the belief $\mathcal{N}(120.0,\ 9.0)$ to $\mathcal{N}(118.5,\ 9.25)$. In words, the level should fall by the draw, $120.0 - 1.5 = 118.5$, and the uncertainty grows by the process noise, $9.0 + 0.25 = 9.25$, which is Equation 7 done by hand for a bell curve. Correct: the sensor reports $z_1$ of 115.0. The innovation is $115.0 - 118.5 = -3.5$ centimetres. Is that the tank or the sensor? Compute the shares. $S$ is $9.25 + 4.0 = 13.25$, so the gain is $9.25/13.25 = 0.698$. Most of the doubt sits in the prediction, because the dipstick read was poor, so the filter believes the sensor more than the model and moves about seventy per cent of the way to it: $118.5 + 0.698 \times (-3.5) = 116.06$, with uncertainty $(1 - 0.698) \times 9.25 = 2.79$.

Notice that it did not move to 115.0. A sensor with $\sigma = 2$ reporting 115.0 is consistent with a true level of 116, and the filter kept the part of the prediction worth keeping.

**Minute 2, pump on**, so $u_k = 1$. Predict: $116.06 + 4.0 - 1.5 = 118.56$, uncertainty 3.04. The sensor reports 121.0, an innovation of 2.44, so $S$ is 7.04 and the gain is $K = 0.432$. The estimate lands at 119.61 and the uncertainty at 1.73. The gain fell from 0.698 without anyone touching it: the filter knows more than it did, and has stopped chasing the sensor as hard.

**Minute 3, pump on.** Predict 122.11, uncertainty 1.98. The sensor reports 122.4, an innovation of only 0.29. The gain is $K = 0.331$, the estimate 122.21, the uncertainty 1.32.

Table 1: Three minutes of the tank filter, with levels in centimetres and $P$ in square centimetres.

| Minute | pump | prediction | sensor | innovation | gain | estimate | $P$ |
|---|---|---|---|---|---|---|---|
| 0 | – | – | – | – | – | 120.00 | 9.00 |
| 1 | off | 118.50 | 115.0 | −3.5 | 0.698 | 116.06 | 2.79 |
| 2 | on | 118.56 | 121.0 | 2.44 | 0.432 | 119.61 | 1.73 |
| 3 | on | 122.11 | 122.4 | 0.29 | 0.331 | 122.21 | 1.32 |

Two patterns are readable off Table 1 that the equations alone do not show. The gain column falls monotonically and the uncertainty falls with it, which is the filter converging from a bad initial guess. And the innovation column shrinks, from 3.5 to 2.44 to 0.29, which is the filter's predictions getting good. Those two columns are the diagnostic pair, and Practical Failure Modes is largely about reading them.

One more thing the table shows by what it omits. After minute 3 the filter holds $\mathcal{N}(122.21,\ 1.32)$ and nothing else. The readings 115.0, 121.0 and 122.4 have been discarded, and no later step will ever ask for them. That is the Markov property cashed out as a memory footprint: a belief about the present state is a sufficient summary of every sample

that went into it, so the past can be thrown away with nothing lost. A twin that logs every reading to a database still does so, for audit and for people; the estimator does not need them.

Run this filter another twenty minutes and the gain stops falling, settling near $K = 0.221$, because $P$ has reached the value where certainty gained from each measurement exactly balances certainty lost to $Q$ in each prediction. That steady-state gain is a property of $Q$ and $R$ alone: halve the sensor noise and it rises, halve the process noise and it falls.

## 7. Second Worked Example: Estimating What Nobody Measures

Return to the building manager, who wanted the draw. The filter above was told the draw was 1.5 and never questioned it. Residents are not so regular.

The technique is **state augmentation**, and its idea is productively disrespectful: if you do not know a quantity, stop treating it as a constant of the model and declare it part of the state. The estimator already infers states from indirect evidence, so give it one more. This is not a trick invented for teaching. A validated digital twin of a floating offshore wind turbine estimates structural states *and* the aerodynamic loads acting on them with exactly this move, augmenting the Kalman filter used for torque estimation with additional states and outputs [8]. The general name for the output is a virtual sensor. Couple an executable model to whatever instruments an asset already carries and the pair yields readings that no instrument on it produces, which is how a short sensor list is made to answer questions it has no channel for [9].

The state becomes $[\, L,\ d \,]^{\mathsf{T}}$, so $n = 2$ while $p = 1$ – one sensor, two unknowns. The draw needs its own transition rule, and the honest one is that it does not change except when it does: a random walk, in which the draw next minute is the draw this minute plus a small disturbance. Augmentation is two Markov chains joined into one whose state is the pair, and the joined chain keeps the Markov property that the level alone had lost. Once the draw is in the state, the level's history has nothing left to say about the future that the present pair does not.

$$\begin{bmatrix} L_{k+1} \\ d_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} L_k \\ d_k \end{bmatrix} + \begin{bmatrix} b \\ 0 \end{bmatrix} u_k + w_k \tag{11}$$

The $-1$ in the top right of Equation 11 is the only interesting entry. It says the draw subtracts from the level, and it is the channel through which the estimator learns about $d$ from observations of $L$. The measurement matrix is the row $[\, 1 \;\; 0 \,]$: the sensor sees the level and is blind to the draw. Process noise keeps 0.25 on the level and takes something small on the draw, say 0.01, stating that the draw wanders slowly.

**Predict**, from $[\, 120.0,\ 1.5 \,]$ with a diagonal covariance holding 9.0 for the level and 0.25 for the draw, pump off. The state goes to $[\, 118.5,\ 1.5 \,]$, since the draw's own rule leaves it alone. The covariance is the part worth following: computing $A\,P\,A^{\mathsf{T}} + Q$ gives 9.5 in the top left, 0.26 in the bottom right, and – this is the point – $-0.25$ in both off-diagonal entries.

Those entries did not exist a moment ago; the prediction created them, because the joined chain's transition rule moves the level by an amount that depends on the draw. They encode a *correlation*: the filter now knows that if the level turns out lower than predicted, the

draw was probably higher than assumed. That inference is the entire mechanism of virtual sensing, and it lives in one number the covariance recursion produced by itself.

**Correct.** The sensor reports 115.0, the innovation is $-3.5$ again, and $S$ works out at 13.5. The gain is a column of two numbers now, about 0.704 for the level and about -0.0185 for the draw. Apply them and the level estimate lands at 116.04 while the draw estimate moves from 1.50 to 1.56. It moved, and nobody measured it: the off-diagonal entry converted the level's surprise into evidence about the draw, and the draw variance has already ticked down from 0.26 to 0.255.

**The step left to you.** Compute $P$ for this step and confirm that figure. The formula is the matrix version in Equation 10, and the answer is a symmetric two-by-two with 2.81 top-left, $-0.074$ off-diagonal and 0.255 bottom-right.

Augmentation has a limit, and it is called **observability**. With the pump off, the draw is the only thing moving the level, so the filter can isolate it; were the pump rate *also* unknown, two unknowns would push the level in opposite directions and one sensor could never say which had changed. The exercises make you find it.

## 8. When Linear and Gaussian Is Not Enough

Real systems are not linear, and the tank stops being linear the moment the draw is modelled properly. Water leaving through an orifice flows faster when the tank is fuller, roughly as $d_k = c\sqrt{L_k}$. No matrix $A$ does that. The general shape replaces the matrices with functions.

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) + w_k \\ z_k &= h(x_k) + v_k \end{aligned} \tag{12}$$

The chain is still a chain. Equation 2 holds exactly as before, and the two steps in Estimation as Predict and Correct are still the right two steps. What breaks is the arithmetic. Push a bell curve through a curved $f$ and what comes out is not a bell curve, so the belief can no longer be carried as a mean and a covariance, and Equation 7 has no closed form. Four families answer Equation 12, each of them one answer to the question *how do I push a belief through a transition rule I cannot integrate?*, and choosing between them is an engineering decision rather than a matter of taste.

**The extended Kalman filter (EKF)** linearises the transition rule. At each step it differentiates $f$ and $h$ at the current estimate to get local matrices $A_k$ and $C_k$, then runs the ordinary Kalman equations with those, so the belief is forced to stay a bell curve. It is cheap and it is the industrial default. Its weakness comes from the approximation: where the function curves sharply across the width of the current uncertainty, the linearisation is taken in the wrong place and the filter can diverge and never recover.

**The unscented Kalman filter (UKF)** refuses to differentiate and samples the transition instead. It picks a small set of representative *sigma points* around the current estimate, pushes each through the true nonlinear function, and reassembles a mean and covariance from where they land. It handles curvature better and needs no derivatives, which matters when $f$ is a simulation you cannot differentiate. It is the most computationally complex member of the Kalman family, propagating seven sigma points in a lithium-ion battery twin that estimates charge, capacity and internal resistance together [3].

**The particle filter (PF)** gives up the Gaussian assumption too, and approximates Equation 7 by brute force. It represents the belief as a cloud of weighted samples, pushes every one through the transition rule, and reweights them by the likelihood in Equation 8, so a sample that explains the reading well gains weight and one that does not loses it. It can represent any distribution, including the multi-modal ones that arise when a system might be in one of several discrete conditions, and it pays in compute: where the UKF propagates seven points, a particle filter may propagate hundreds per step [3]. It also handles a state that is not a number at all. In the uncrewed aerial vehicle (UAV) twin discussed below the state is a damage condition with finitely many values, so its transition rule is a transition matrix of the kind The State-Space Picture began with, and the twin takes the probability of damage progressing in each region as known and fixed given the load on the wing [7]. A Kalman filter has nothing to say about such a state; a particle filter treats it like any other, and each strain measurement is assimilated iteratively through one [7].

A word on names, because both will appear in any search of the literature. The particle filter is a Monte Carlo method applied to a Markov chain. It is not **Markov chain Monte Carlo**, a different family in which a Markov chain is *constructed* as a sampling device to explore one fixed posterior; the twinning literature lists that family among the non-filtering alternatives to recursive estimation [3]. Only the first is a state estimator.

**The moving-horizon estimator (MHE)** gives up the recursion. Rather than carrying one belief forward a step at a time, it keeps the last several measurements in a window and solves an optimisation at every step for the trajectory that best explains all of them, which lets it enforce hard constraints that no Kalman variant can: a level cannot be negative, a concentration cannot exceed one. It is the second of the two main method families named for state estimation in deployable twins, both resting on a weighted combination of measurements and model predictions [9], and the most expensive, because each step is an optimisation rather than a formula.

Table 2: The four families and the assumption each one gives up.

| Method | Assumes | Cost per step | Reach for it when |
|---|---|---|---|
| Kalman filter | linear, Gaussian | lowest | the model really is linear |
| Extended Kalman filter | mild nonlinearity, Gaussian | low | $f$ is smooth and differentiable |
| Unscented Kalman filter | nonlinear, Gaussian | moderate | curvature is real, or $f$ is a black box |
| Particle filter | nothing much | high | the posterior is not bell-shaped |
| Moving horizon | nonlinear, constrained | highest | the state has hard physical bounds |

Read Table 2 down the middle column and the trade is plain: every step away from the linear-Gaussian case is bought with computation. Read it down the left and the assumptions weaken one at a time, which makes the ordering a debugging order – start at the top and move down only when the row above demonstrably fails. The first line is the sharpest and the one beginners cross without noticing: only a linear system can use a Kalman filter, and

meeting a nonlinear one requires a nonlinear estimator [10]. The Gaussian assumption is the softer second line, called rather restrictive and sometimes undesirable in the battery literature, with the particle filter the usual escape [11]. Twin-building tutorials present the same ladder in the same order [5].

## 9. What the Estimator Gives the Twin

Four services come out of one estimator, and they are worth separating because they fail differently.

**A clean, current state.** The dashboard plots $\hat{x}$ rather than $z_k$, quieter than the sensor without being later than it. This is also what makes multiple sensors tractable: where a naive approach averages similar sensors by accuracy, the filter fuses dissimilar ones – a flow meter, a level sensor, a pump current – because each enters through its own row of $C$ and its own entry of $R$ [6].

**Virtual sensors.** The augmented draw estimate is a reading from an instrument nobody installed, and once the habit forms the applications multiply: bearing temperature inside a sealed motor, stress at a weld no gauge can reach, the state of charge of a battery, which is not measurable at all but an inference from voltage and current. These are the otherwise unmeasurable quantities an executable model plus a limited sensor set is deployed to deliver [9].

**Anomaly and attack detection, at no extra cost.** The estimator computes an innovation every step, and in normal operation it is small and random. When the physical system does something the model does not describe – when the chain it is actually following is not the chain the filter assumed – the innovation stops being small and stops being random, and does so *before* any threshold on the raw signal is crossed. The mechanism has been demonstrated directly: in a twin of an incubator, opening the lid of the running device put the system outside the closed-lid model used for estimation, the Gaussian process-noise assumption stopped holding, the filter's predictions stopped agreeing with the sensor data, and that disagreement is the detection [2]. The same residual serves security with no change to the mathematics: attack-detection schemes compute a residual as the difference between the reported output and the value a dynamic state estimator such as a Kalman filter expected [12]. A tampered sensor feed and a stuck valve look different to a threshold and identical to a residual.

To make this quantitative, compare each innovation against the uncertainty the filter declared for it. The filter says the innovation should have covariance $S$, so the normalised quantity below should average about $p$, the number of measurements, if the filter's model of the world is right.

$$\varepsilon_k = y_k^{\top} S_k^{-1} y_k \tag{13}$$

Alarm on Equation 13 rather than on the raw signal and the threshold means the same thing at every operating point, which a threshold on level or temperature never does.

**A starting point for prediction.** To simulate forward – what will this tank do for two hours, will this turbine survive the coming storm – you need an initial condition, and the true state is not available. The estimate is. A forecast is then the chain run with its correct

step switched off, prediction after prediction, the belief spreading by $Q$ at every step exactly as the three-state belief spread under $T$, so forecast quality is bounded by the estimate it started from, which is why prediction, planning and learning are listed downstream of data assimilation and state estimation in formal accounts of what a twin must do [7].

## 10. Five Systems That Work This Way

Five real cases, spread across scale and domain, each named with what its estimator actually estimates.

**A laboratory incubator.** The smallest and most instructive, because its whole twin is small enough to read. The monitoring service is implemented as a Kalman filter, and its authors single the estimator out as the service the rest of the twin leans on [5]. Two of the four uses listed above appear there as deployed features rather than as this chapter's argument. It recovers the temperature *inside* the box, which the model needs and no sensor reads, from the air temperature that is measured. The anomaly demonstration is the opened lid described above [2].

**A floating offshore wind turbine.** At the other end of the scale, and validated against a full-scale prototype rather than a simulation. The twin uses linearised turbine models with a Kalman filter at the core of its state estimation, augmented to recover structural states and aerodynamic loads together [8]. Loads on a rotor in open water are the archetypal unmeasurable quantity: power and rotor speed are cheap to measure, and a load cell on a blade in the North Sea is not.

**A UAV airframe.** Here the estimated state is *structural health* – how damaged the aircraft is. The damage state takes finitely many values, so the chain is the discrete kind with a transition matrix rather than a transition density, and its entries encode how likely damage is to progress in each region under a given load [7]. Strain sensors report noisily, the twin carries a sensor model that deliberately does not match reality exactly, and a particle filter assimilates each measurement to maintain a posterior over the damage state, which then feeds mission planning [7]. This is where the Gaussian assumption genuinely fails, because a belief over discrete damage conditions is not bell-shaped.

**A railway bridge.** Similar in spirit, different in tempo. The digital state comprises structural health parameters, one noisy observation is assimilated per time step, and the twin re-estimates the damage parameters every time a train crosses [13]. The update rate is tied to the physical event rate rather than to a clock, a useful reminder that "once per minute" was a choice in our tank and not a law. The same work notes that the chain's transition model is bound up with how the digital state is parametrised and with what historical data exists to fit it [13], which is the choice-of-state question from The State-Space Picture arriving in a civil-engineering office.

**A lithium-ion battery pack.** The digital state is the set of parameters characterising the cell's model, with state of charge and state of health the quantities of interest, neither measurable with an instrument [3]. Fade curves bend, which rules the plain Kalman filter out and puts the battery literature's usual pick in the second and third rows of our table, with the particle filter held back for the cases a bell curve cannot describe [11]. Every phone and electric vehicle you have used runs an estimator of this kind and reports its output as a percentage.

Medical twins deserve a closing sentence for contrast, since they solve this problem offline to personalise themselves and then update continually as data arrives [14]. Same machinery, different clock.

## 11. Practical Failure Modes

Getting the equations right is the half a textbook can teach. These are the four ways a correct implementation still misbehaves.

**Tuning $Q$ and $R$.** $R$ is the measurable one: leave the sensor pointing at something stationary for an hour and take the variance of what it reports. $Q$ is not, because it stands for the model's error, and if you could measure that you would fix the model instead. In chain terms $Q$ is the spread of each transition arrow, the filter's statement of how far the system may wander from where the model says in one step. Treat $Q$ as the one honest knob. Too small and the filter over-trusts the model, ignores measurements and drifts away from reality while reporting high confidence – the dangerous failure, because the twin is both wrong and sure. Too large and the filter chases noise, and you have paid for a Kalman filter to get a sensor. The diagnostic is Equation 13: a running average far above $p$ means $Q$ or $R$ is too small, far below means one is too large. Neither number should be guessed and then left alone: the offshore-turbine work tabulated how estimation error rose as it added noise proportional to each input's own standard deviation, presenting that as a rough quantification of what to expect in the field [8], and running the same experiment on your own system before deployment converts an argument into a number.

**Unobservability.** Some states cannot be recovered from the available measurements however good the filter, and for a linear system this is decidable in advance. Stack the measurement matrix against powers of the process matrix,

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

and if that matrix has rank less than $n$, some direction in the state space is invisible and the filter's uncertainty in it never shrinks. A covariance that refuses to converge in one component is usually saying this, and the fix is another sensor, not another hour of tuning.

**Model mismatch.** The filter assumes its transition density and its likelihood are the right ones; they are not, and the mature response is to say so in the model rather than hope. The UAV twin's sensor model is described as deliberately not matching the simulated measurements exactly [7] – a filter tested only against data its own model generated has not been tested. Inflating $Q$ is the standard blunt remedy.

**Time, which textbooks usually omit.** A twin is a networked system, and its measurements arrive late, out of order, and stamped by a clock that is not yours. The chain is indexed by step, and Equation 8 applies a reading to the state at the step it belongs to. A filter fed a measurement at the wrong step applies it to the wrong link of the chain and corrects a prediction that was never wrong, which shows up as a persistent oscillation no tuning removes. Twin architectures make the update rate an explicit design parameter for

this reason [13], and delay and missing signals sit alongside noise as the obstacles a twin's estimator must overcome [2]. The remedies are ordinary engineering: timestamp at the sensor rather than the broker, buffer briefly and process in timestamp order, and skip the correct step rather than fabricate a measurement when nothing arrives.

## 12. Exercises

The objective each exercise serves is given in brackets; hints and short solutions follow the set.

1. *(Objective 1.)* A greenhouse is heated by a radiator and loses heat to the outside. Write a scalar model in the shape of Equation 3 with internal temperature as the state. Say what is in $u_k$, and name two physical effects you put into $w_k$.
2. *(Objective 2.)* Using the three-state transition matrix $T$ from The State-Space Picture, start certain that the tank is *high*, $\pi_0 = [\,0,\ 0,\ 1\,]$, and compute the belief after one step and after two. Then say, without computing further, what the belief tends to as the steps go on, and why a measurement is the only thing that can reverse the trend.
3. *(Objective 2.)* Two proposed states for the tank, and whether each keeps the promise in Equation 2. First, a colleague suggests storing the level *rounded to the nearest ten centimetres*, to save bandwidth. Explain why that state is not Markov. Second, residents' draw is drifting slowly and the state is the exact level alone. Explain what the level's recent history now says about the future that the present level does not, and name the repair.
4. *(Objective 3.)* Continue the tank filter for a fourth minute: pump on, sensor reports 125.1. Compute prediction, innovation, gain, estimate and covariance, and write the prediction as a belief in the $\mathcal{N}(\text{mean},\ \text{variance})$ form the Worked Example used.
5. *(Objective 3.)* The sensor goes offline for ten minutes while the pump runs. Describe what the filter outputs during the outage in terms of the two chain equations, and what happens to $P$. What is the gain on the first measurement after the link returns, and why is that right?
6. *(Objective 4.)* In Equation 11, suppose the pump rate $b$ were also unknown and added to the state, giving three states and still one sensor. Argue, without computing the observability matrix, why the draw and the pump rate cannot both be recovered while the pump runs. Then describe an operating schedule that makes them recoverable.
7. *(Objective 5.)* The draw is modelled properly as $d_k = c\sqrt{L_k}$. Derive the scalar Jacobian an extended Kalman filter would need for the level equation. At which level is the linearisation worst, and what does that suggest about a nearly empty tank?
8. *(Objective 6.)* A filter has run well for a month. This week its innovations have been small but consistently positive. Separately, a colleague's filter shows innovations that are large but average to zero. Diagnose both, and say which is more urgent.

**Hints and short solutions.** (1) State is internal temperature, $u_k$ the radiator command; $w_k$ absorbs door openings and solar gain. (2) After one step $[\,0,\ 0.10,\ 0.90\,]$; after two $[\,0.005,\ 0.18,\ 0.815\,]$. The belief tends to $[\,0.25,\ 0.50,\ 0.25\,]$, the one row that $T$ leaves unchanged, and every multiplication by $T$ moves towards it and never away, because prediction only ever averages; Equation 8 is the step that can sharpen a belief, and it needs a reading to do it. (3) Two tanks both rounded to 120 may really stand at 115.1 and 124.9, and the true level, which recent readings hint at, decides which bin comes next, so the past carries information the rounded present has thrown away. In the second case a level that has been

falling faster than the model expects says the draw has risen, and so says the level will keep falling faster; the present level alone cannot. The repair in both cases is a bigger state – the exact level in the first, the pair of level and draw in the second. (4) Prediction 124.71, so the predicted belief is $\mathcal{N}(124.71,\ 1.57)$; innovation 0.39, gain about 0.283, estimate 124.82, covariance about 1.13. (5) Pure model prediction, which is Equation 7 applied ten times with Equation 8 never applied, so $P$ grows by $Q$ every step; the gain on the returning measurement is large, which is right, because ten minutes of unchecked prediction is genuinely uncertain. (6) While the pump runs, a rise in $b$ and a fall in $d$ produce an identical level trajectory, so one sensor cannot separate them; cycling the pump off makes $d$ observable alone, after which $b$ follows. (7) The Jacobian is $\dfrac{c}{2\sqrt{L_k}}$, which blows up as the level approaches zero, so the linearisation is worst in a nearly empty tank – an argument for a UKF or a floor on the model. (8) A biased innovation means a model error, typically an unmodelled drift, and it is the urgent one because the filter is confidently wrong; large zero-mean innovations mean $R$ is understated or the sensor has degraded, which is visible and less dangerous.

## 13. The Plant and the Controller

Control engineering has had this chapter's machinery for sixty years under its own names, and most of the twinning literature is written in them. The body of the chapter kept that vocabulary out so the estimator could be learned on its own terms; collected here, the analogy is worth having before you read further.

**Plant, controller, observer.** In control theory the system being acted on is the *plant*, the algorithm choosing its inputs is the *controller*, and the algorithm that reconstructs the plant's state from its outputs, so that the controller has something to act on, is the *observer*. Every estimator in this chapter is an observer, and the Kalman filter is the observer for the linear-Gaussian plant. In the chain vocabulary of The State-Space Picture, the plant *is* the hidden Markov chain and the observer is whatever maintains a belief about its present state. The loop figure in Why a Twin Needs a State Estimator is a control loop drawn with the observer made large: the real system is the plant, the box marked control, alarms and forecasts is the controller, and the dashed actuation edge is the controller's output returning to the plant.

**The twin as the controller's side of the loop.** What separates a twin from a dashboard is that information travels both ways, and authoritative accounts of the field describe the inbound direction as continuously revising the model from data and the outbound one as turning the revised picture into choices about what to do and what to measure next [15]. In control terms the inbound half is the observer and the outbound half is the controller, and the loop that defines a twin closes where the estimate becomes the input to a decision; the estimator is its first half. Set asset and twin side by side as two coupled dynamical systems, each evolving through its own state space, and the estimator is the arrow between them: the channel by which the twin learns the asset's state so that it can in turn steer the asset towards favourable states [7].

**The plant model, on both sides.** The attack-detection residual of What the Estimator Gives the Twin is computed by a filter that uses the same system, input and output matrices as the plant model [12]. Plant and observer share $A$, $B$ and $C$, which is why a disagreement between them says that the *plant* has changed, not the model. That the same matrices appear on both sides of the loop is the whole content of the phrase *model-based*: the observer

is the plant's own model, run alongside the plant and corrected by it.

**Where the analogy stops.** A controller's observer is usually designed once and left alone. A twin's estimator is expected to be recalibrated, to have its $Q$ and $R$ revisited as Practical Failure Modes describes, and to report its own confidence to people as well as to a control law. The vocabulary transfers; the operating assumptions do not, and the failure modes above are mostly what happens in the gap.

## 14. Summary and Where to Go Next

A digital twin holding only sensor data is missing its most useful component. Three gaps motivated the fix – noise, unmeasured quantities and missing samples – and one mechanism closed all three: a running belief about the present state of a Markov chain, carried forward through the chain's transition rule and corrected by Bayes' rule at each measurement, with the weight between them set by which is currently more trustworthy. The Kalman filter is that mechanism in the one case where the arithmetic closes, and every method in When Linear and Gaussian Is Not Enough is the same mechanism where it does not.

Against the objectives: you have a state-space model for a tank; you can state the Markov property, have watched a three-state chain lose certainty under it and have seen it fail for a badly chosen state; you have run a scalar Kalman filter by hand, read each step as a chain operation, and watched its gain fall from 0.698 to 0.331 without intervention; you have augmented a state to recover a draw nobody measures, and seen the off-diagonal covariance entry that makes it possible; you have a table sorting four method families by the assumption each gives up; and you have four failure modes with a diagnostic each.

**Deeper into the mathematics.** The two chain equations were stated for the continuous case and worked by hand only for a three-state table; deriving the Kalman prediction step from Equation 7 for a Gaussian belief is a page of algebra and the best single exercise for seeing why the filter has the shape it has. The gain was stated, never derived; choosing $K$ to minimise the trace of the posterior covariance is short and worth doing once. Smoothing is what you want for forensic analysis rather than live control.

**Deeper into the twinning literature.** The incubator tutorial [5] is the place to see an estimator inside a complete twin, with the technical treatment of its Kalman filter [2] as companion. For the formal view, the probabilistic graphical model account [7] is the reference, with a civil-engineering instantiation available separately [13]. Where the question is deployment rather than formalism, the executable digital twin treatment [9] places estimation alongside model order reduction and co-simulation.

**Wider.** Two surveys give the map: a two-part review of twin enabling technologies with estimation and uncertainty quantification as its spine [3], [11], and a systematic review of predictive maintenance built on twins [10]. For what changes when a twin becomes a population rather than one instrument, the industrial-scaling account [14] is the honest one. The Markov chain that has been a mathematical convenience here becomes a design object once the twin *chooses* the inputs, which is the territory of Markov decision processes; the formal account of twins draws on the partially observable version of that framework for its coupled asset-and-twin model [7]. One direction left out entirely is learned estimators, where a neural network replaces the process model or the whole filter: promising, not yet the default, and everything here about innovations and covariances is the right vocabulary

for judging whether one is working.

## 15. References


[1] C. Gomes, H. Feng, Z. Kazemi, and K. Pierce, "Calibration of Models for Digital Twins," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 129–146. doi: 10.1007/978-3-031-66719-0_6.

[2] 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.

[3] A. Thelen *et al.*, "A comprehensive review of digital twin — part 1: Modeling and twinning enabling technologies," *Structural and Multidisciplinary Optimization*, vol. 65, no. 12, p. 354, Nov. 2022, doi: 10.1007/s00158-022-03425-4.

[4] G. Abbiati, C. Gomes, M. Sandberg, Z. Kazemi, S. T. Hansen, and P. G. Larsen, "Modelling for Digital Twins," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 89–127. doi: 10.1007/978-3-031-66719-0_5.

[5] C. Gomes *et al.*, "Digital Twin Tutorial: The Incubator Case Study," in *Engineering Trustworthy Software Systems: 6th International School, SETSS 2024, Chongqing, China, April 14–21, 2024, Tutorial Lectures*, J. P. Bowen, C. Gomes, and Z. Liu, Eds., Singapore: Springer Nature, 2025, pp. 68–101. doi: 10.1007/978-981-96-4656-2_3.

[6] C. Gomes, D. E. L. Rötter, A. Iosifidis, H. Feng, H. Ejersbo, and M. Frasheri, "Sensing and Communication of Data from the Physical Twin," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 147–171. doi: 10.1007/978-3-031-66719-0_7.

[7] 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.

[8] 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.

[9] D. Hartmann and H. Van der Auweraer, *The Executable Digital Twin: Merging the digital and the physics worlds*. 2022.

[10] R. van Dinter, B. Tekinerdogan, and C. Catal, "Predictive maintenance using digital twins: A systematic literature review," *Information and Software Technology*, vol. 151, p. 107008, Nov. 2022, doi: 10.1016/j.infsof.2022.107008.

[11] A. Thelen *et al.*, "A comprehensive review of digital twin—part 2: Roles of uncertainty quantification and optimization, a battery digital twin, and perspectives," *Structural and Multidisciplinary Optimization*, vol. 66, no. 1, p. 1, Dec. 2022, doi: 10.1007/s00158-022-03410-x.

[12] T. Kulik, Z. Kazemi, and P. G. Larsen, "Security and Privacy-related Issues in a Digital Twin Context," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 313–344. doi: 10.1007/978-3-031-66719-0_13.

[13] 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.

[14] S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, "Scaling digital twins from the artisanal to the industrial," *Nature Computational Science*, vol. 1, no. 5, pp. 313–320, May 2021, doi: 10.1038/s43588-021-00072-5.

[15] *Foundational Research Gaps and Future Directions for Digital Twins*. Washington, D.C.: National Academies Press, 2024. doi: 10.17226/26894.