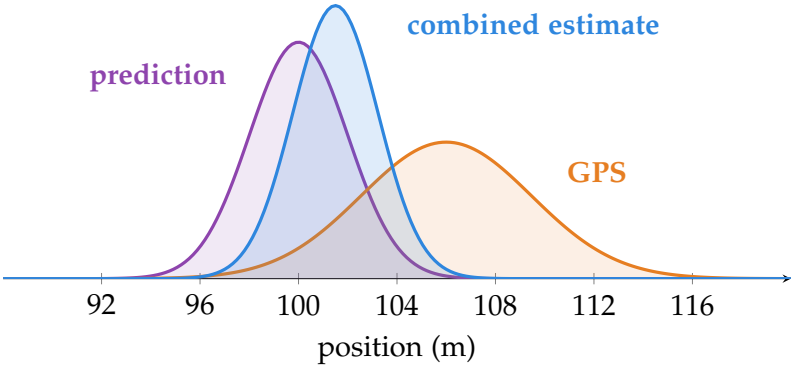
The Kalman Filter

How two imperfect sources make one better estimate



Detailed study notes with figures, worked examples
and complete mathematical solutions

Based on the video *"Kalman Filters"* (car-and-GPS example),
expanded with derivations, extra examples and practice problems.

Colour code used throughout: **purple = prediction (model)**, **orange = measurement (GPS)**, **blue = Kalman estimate**.

Contents

1 The problem: two imperfect sources **3**

2 Describing uncertainty with a bell curve **4**

2.1 The state . . . 4

2.2 Mean and variance . . . 4

2.3 Two useful facts about variance . . . 4

3 The worked example from the video **5**

3.1 The setup . . . 5

3.2 The Kalman gain . . . 5

3.3 The new estimate . . . 5

3.4 The new uncertainty . . . 6

4 Where does the gain come from? (Full derivation) **6**

4.1 What the filter is allowed to choose . . . 6

4.2 Step 1: write the error of the estimate . . . 7

4.3 Step 2: compute its variance . . . 7

4.4 Step 3: find the lowest point . . . 8

4.5 Step 4: the smallest variance . . . 8

4.6 Step 5: precisions add . . . 8

4.7 Step 6: sanity checks (limiting cases) . . . 9

5 The predict–update loop **9**

5.1 The prediction step . . . 10

5.2 The update step . . . 10

5.3 Watching the filter settle (full calculation) . . . 10

5.4 The steady state, calculated exactly . . . 11

6 More than one number: position and velocity **12**

6.1 The state vector and the covariance matrix . . . 12

6.2 The matrices of the model . . . 13

6.3 The matrix equations . . . 13

6.4 Why $FPF^\top$? (Gaussian through a linear map) . . . 13

6.5 The worked 2-D example from the video . . . 14

6.6 One more cycle (our own extension) . . . 16

6.7 Real-world versions . . . 17

7 Derivation of the matrix Kalman gain (optional) **17**

8 Limits and variations **18**

8.1 Assumption 1: the system is linear 18

8.2 Assumption 2: the noise levels are right 18

9 Practice problems with full solutions **20**

10 Summary cheat sheet **22**

A Python code for the examples **23**

1 The problem: two imperfect sources

In 1968 the Apollo 8 astronauts flew to the Moon with a computer that had only a few kilobytes of working memory. The computer had to know *where the spacecraft was* at every moment. It had two ways to find out, and neither one was good enough alone.

1. **Physics (a model).** If you know where you were and how fast you were moving, you can calculate where you are now. But a tiny error in one calculation is carried into the next one. The errors pile up, and the estimate slowly *drifts* away from the truth.

2. **Measurement (a sensor).** The astronauts looked at stars with a sextant. A measurement does not pile up errors over time, because each sighting is tied to the real world. But every sighting is *noisy*: the readings jump around.

The navigation software combined the two, and the result was more accurate than either one. Your phone does the same thing today: it mixes a model of how you move with noisy GPS readings, so the blue dot on the map does not jump around. This method is called the **Kalman filter**.

Figure 1 shows the idea with a simulated car. The orange line (GPS only) jitters. The purple line (model only) drifts far away. The blue line (Kalman filter) stays close to the truth.

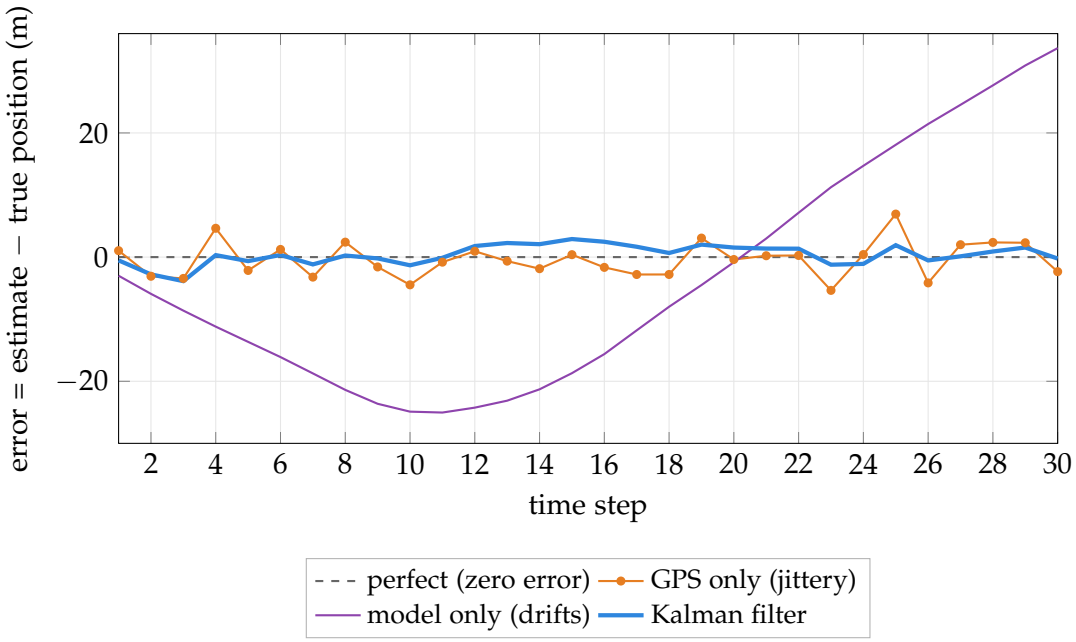


Figure 1: A simulated car over 30 seconds. In this run the GPS error has a typical size of about 2.8 m, while the Kalman filter error is about 1.6 m. The model alone was started with a slightly wrong speed and drifts by tens of metres.

Key idea

Trusting only the sensor gives a *jittery* estimate. Trusting only the model gives a *drifting* estimate. A fixed 50/50 average only makes sense if both are equally reliable, which is rarely true. We need a **weight that reflects how much each source deserves to be trusted right now**. The Kalman filter computes exactly that weight.

2 Describing uncertainty with a bell curve

2.1 The state

The **state** is the set of numbers that describes the system. For a car on a straight road, the state could simply be its position x. In a more realistic case it could also contain velocity and acceleration. We never see the state perfectly; we only have the model and the sensor.

2.2 Mean and variance

Instead of storing one number ("the car is at 100 m"), the Kalman filter stores a whole **bell curve** (a Gaussian, or normal distribution):

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

- The **mean** μ is the centre of the curve: our best guess.
- The **variance** σ^2 is the spread: how unsure we are. A big variance means a wide, flat curve.
- The **standard deviation** $\sigma = \sqrt{\sigma^2}$ is in the same units as the state (metres). About 68% of the time the true value lies within $\pm\sigma$ of the mean, and about 95% of the time within $\pm 2\sigma$.

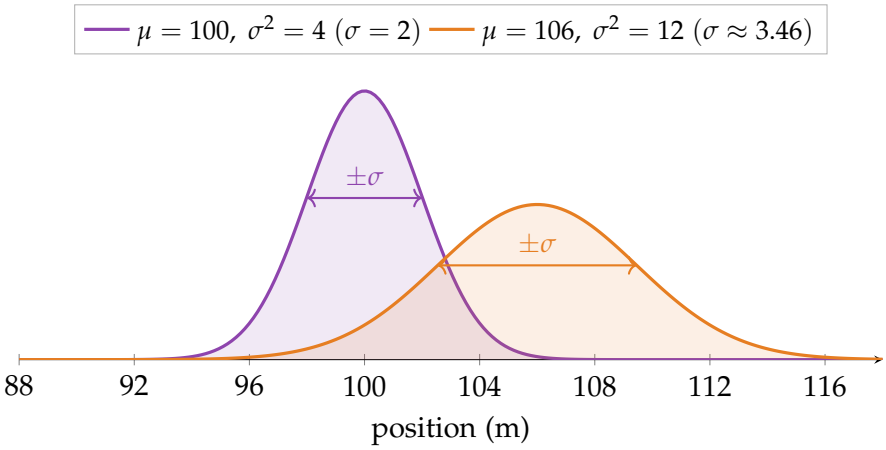


Figure 2: Two bell curves from the video. The prediction (purple) is narrow: we are fairly sure. The GPS (orange) is wide: it is less precise.

In plain words

Variance 4 means "usually within about 2 m" (because $\sqrt{4} = 2$). Variance 12 means "usually within about 3.5 m" (because $\sqrt{12} \approx 3.46$). Once each source is a bell curve, the problem becomes: *take two uncertain estimates and combine them into a better one.*

2.3 Two useful facts about variance

We will use these two facts many times. Let a, b be constants and X, Y be random errors.

$$\mathrm{Var}(aX) = a^2 \,\mathrm{Var}(X), \tag{1}$$

$$\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) \quad \text{if } X \text{ and } Y \text{ are independent.} \tag{2}$$

Fact (1) says: if you multiply an error by a, its variance is multiplied by a^2 (squares, because variance is measured in "metres squared"). Fact (2) says: variances of independent errors simply add.

3 The worked example from the video

3.1 The setup

A car drives along a straight road.

- The **model** uses the old position, the speed and the time that has passed, and predicts that the car is at $x_{\text{pred}} = 100$ m with variance $\sigma^2_{\text{pred}} = 4$.
- At the same moment the **GPS** says the car is at $z = 106$ m, with variance $\sigma^2_{\text{GPS}} = 12$.

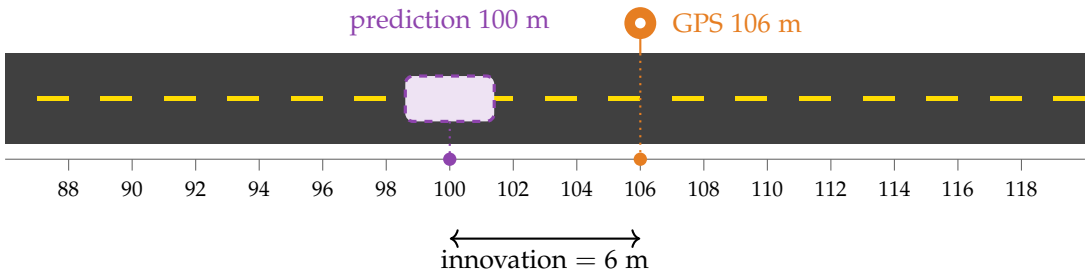


Figure 3: The prediction and the GPS disagree by 6 m.

The difference between what we measured and what we predicted is called the **innovation** (or simply the *surprise*):

$$y = z - x_{\text{pred}} = 106 - 100 = 6 \text{ m}.$$

The big question is: *how much of this 6 m surprise should we believe?*

3.2 The Kalman gain

The Kalman filter answers with a number K between 0 and 1, called the **Kalman gain**. It is the fraction of the surprise that we accept. For one-dimensional problems:

Kalman gain (1-D)

$$K = \frac{\sigma^2_{\text{pred}}}{\sigma^2_{\text{pred}} + \sigma^2_{\text{GPS}}}$$

For our numbers:

$$K = \frac{4}{4+12} = \frac{4}{16} = \frac{1}{4} = 0.25.$$

3.3 The new estimate

We move one quarter of the way from the prediction towards the measurement:

$$\hat{x} = x_{\text{pred}} + K\,(z - x_{\text{pred}}) = 100 + \tfrac{1}{4} \cdot 6 = 100 + 1.5 = 101.5 \text{ m}.$$

The same formula can be written as a weighted average:

$$\hat{x} = (1 - K)\, x_{\text{pred}} + K\, z = \tfrac{3}{4} \cdot 100 + \tfrac{1}{4} \cdot 106 = 75 + 26.5 = 101.5 \text{ m}.$$

Both forms are identical: expand $(1 - K)x_{\text{pred}} + Kz = x_{\text{pred}} + K(z - x_{\text{pred}})$.

The estimate stays close to the prediction, because in this case the prediction was the more reliable source (variance 4 versus 12).

3.4 The new uncertainty

The new variance is

$$\sigma^2_{\text{new}} = (1 - K)\, \sigma^2_{\text{pred}} = \tfrac{3}{4} \cdot 4 = 3.$$

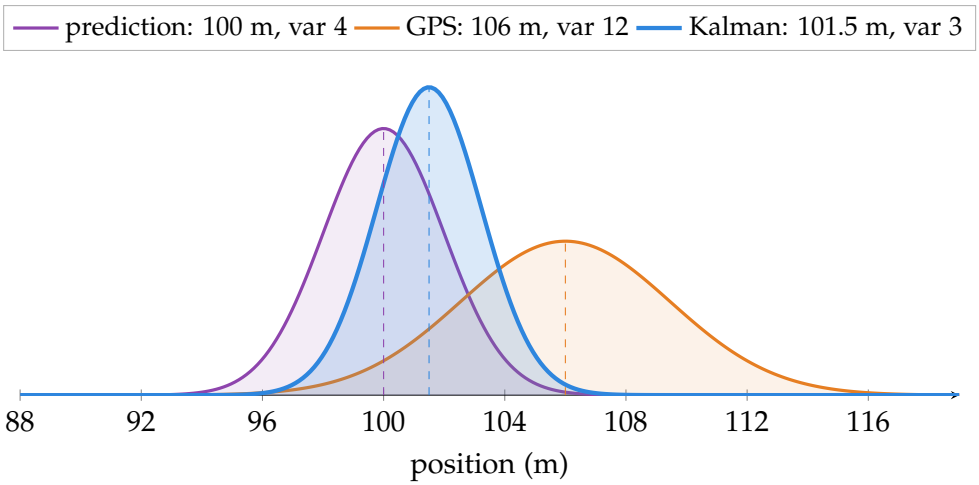


Figure 4: After the update the blue curve is *taller and narrower* than both inputs: it is more certain than either source.

The surprising result

The new variance 3 is smaller than the prediction's 4, and much smaller than the GPS's 12. By combining a fairly good source with a worse one, we get an estimate that is **better than the good source alone**. The rest of this section explains why.

4 Where does the gain come from? (Full derivation)

4.1 What the filter is allowed to choose

The new estimate is a mixture of prediction and measurement:

$$\hat{x} = (1 - K)\, x_{\text{pred}} + K\, z.$$

The weights $(1 - K)$ and K add up to 1. This matters: if both sources are right on average, the mixture is also right on average (it is *unbiased*). Every choice of K gives a different estimate, and also a different amount of leftover uncertainty. **The goal is to pick the K that makes the leftover uncertainty as small as possible.**

4.2 Step 1: write the error of the estimate

Let x be the (unknown) true position. Write each source as "truth + error":

$$x_{\text{pred}} = x + e_p, \qquad z = x + e_g,$$

where e_p has variance σ_p^2 (prediction) and e_g has variance σ_g^2 (GPS). We assume the two errors are **independent**. Then

$$\begin{aligned}\hat{x} &= (1-K)(x+e_p) + K(x+e_g) \\ &= \underbrace{(1-K)x + Kx}_{=x} + (1-K)e_p + Ke_g,\end{aligned}$$

so the error of our estimate is

$$\hat{x} - x = (1-K)\,e_p + K\,e_g.$$

4.3 Step 2: compute its variance

Using facts (1) and (2):

Variance of the combined estimate

$$\sigma_{\text{new}}^2(K) = (1-K)^2\,\sigma_p^2 + K^2\,\sigma_g^2.$$

In words: the prediction's variance is multiplied by the *square* of the share we kept from the prediction, and the sensor's variance is multiplied by the *square* of the share we took from the sensor.

For the car: $\sigma_{\text{new}}^2(K) = 4(1-K)^2 + 12K^2$.

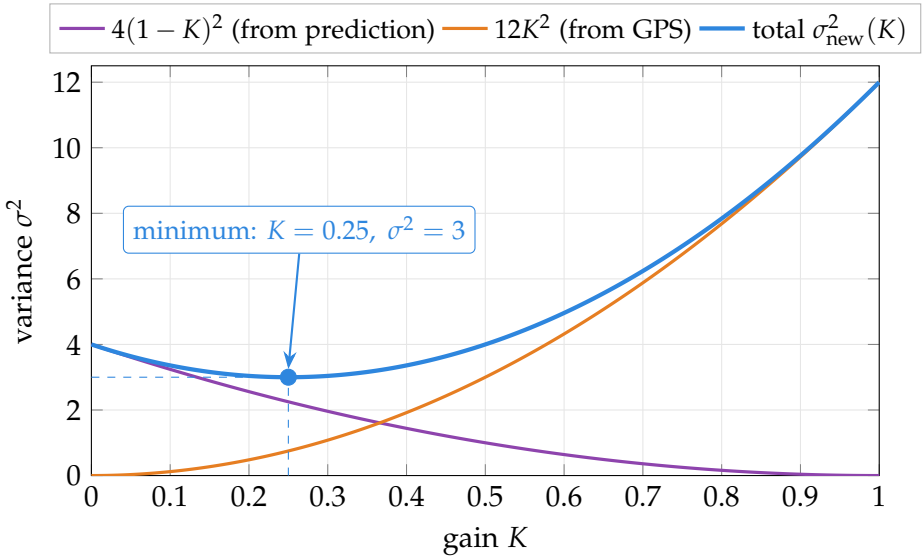


Figure 5: As K grows, the purple part shrinks and the orange part grows. The total (blue) has a lowest point at $K = 1/4$.

4.4 Step 3: find the lowest point

At the minimum, the slope is zero. Differentiate with respect to K:

$$\frac{d\sigma^2_{\text{new}}}{dK} = 2(1-K)(-1)\,\sigma_p^2 + 2K\,\sigma_g^2 = -2(1-K)\sigma_p^2 + 2K\sigma_g^2.$$

Set it equal to zero and solve:

$$\begin{aligned} -2(1-K)\sigma_p^2 + 2K\sigma_g^2 &= 0 \\ K\sigma_g^2 &= (1-K)\sigma_p^2 \\ K\sigma_g^2 + K\sigma_p^2 &= \sigma_p^2 \\ K &= \frac{\sigma_p^2}{\sigma_p^2 + \sigma_g^2}. \end{aligned}$$

This is exactly the gain we used. Is it really a minimum and not a maximum? The second derivative is

$$\frac{d^2\sigma^2_{\text{new}}}{dK^2} = 2\sigma_p^2 + 2\sigma_g^2 > 0,$$

so the curve bends upward (like a bowl) and the point is a minimum.

Check with the car: $\frac{d}{dK}\left[4(1-K)^2 + 12K^2\right] = -8(1-K) + 24K = 32K - 8 = 0 \Rightarrow K = \frac{8}{32} = \frac{1}{4}$. ✓

4.5 Step 4: the smallest variance

Plug the best K back in. Note that $1 - K = \dfrac{\sigma_g^2}{\sigma_p^2 + \sigma_g^2}$. Then

$$\sigma^2_{\text{new}} = \left(\frac{\sigma_g^2}{\sigma_p^2+\sigma_g^2}\right)^2 \sigma_p^2 + \left(\frac{\sigma_p^2}{\sigma_p^2+\sigma_g^2}\right)^2 \sigma_g^2 = \frac{\sigma_p^2\sigma_g^2\,(\sigma_g^2+\sigma_p^2)}{(\sigma_p^2+\sigma_g^2)^2} = \frac{\sigma_p^2\,\sigma_g^2}{\sigma_p^2+\sigma_g^2}.$$

This equals $(1-K)\sigma_p^2$, the short formula we used earlier. For the car: $\dfrac{4\cdot 12}{4+12} = \dfrac{48}{16} = 3$. ✓

4.6 Step 5: precisions add

Take the reciprocal of the last result:

Precision form

$$\frac{1}{\sigma^2_{\text{new}}} = \frac{\sigma_p^2+\sigma_g^2}{\sigma_p^2\sigma_g^2} = \frac{1}{\sigma_p^2} + \frac{1}{\sigma_g^2}.$$

The number $1/\sigma^2$ is called **precision**. So the rule is: **precisions add**. For the car:

$$\frac{1}{4} + \frac{1}{12} = \frac{3}{12} + \frac{1}{12} = \frac{4}{12} = \frac{1}{3} \quad\Longrightarrow\quad \sigma^2_{\text{new}} = 3.$$

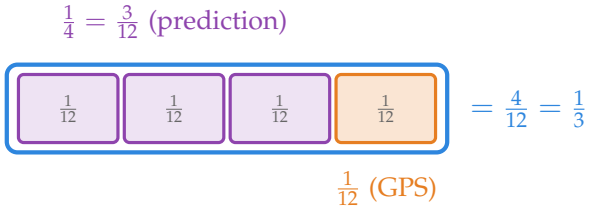


Figure 6: Precision blocks: the prediction contributes 3 blocks, the GPS 1 block. Together: 4 blocks of $1/12$, i.e. $1/3$.

In plain words

Every source of information adds some precision. Even a bad sensor adds a little. Adding precision can never make you *less* precise. That is why combining a good source with a worse one still beats the good source alone (as long as their errors are independent).

4.7 Step 6: sanity checks (limiting cases)

- **Perfect sensor** ($\sigma_g^2 = 0$): $K = \sigma_p^2/\sigma_p^2 = 1$. We jump all the way to the measurement. Correct!
- **Useless sensor** ($\sigma_g^2 \to \infty$): $K \to 0$. We ignore the measurement. Correct!
- **Perfect prediction** ($\sigma_p^2 = 0$): $K = 0$. No reason to change our estimate.
- **Equally good sources** ($\sigma_p^2 = \sigma_g^2$): $K = \frac{1}{2}$, a simple average, and the variance is halved.

Everything in between is weighted according to how certain each source is.

5 The predict–update loop

One update is not enough: the car keeps moving, so the filter repeats the process forever while its own uncertainty changes with time.

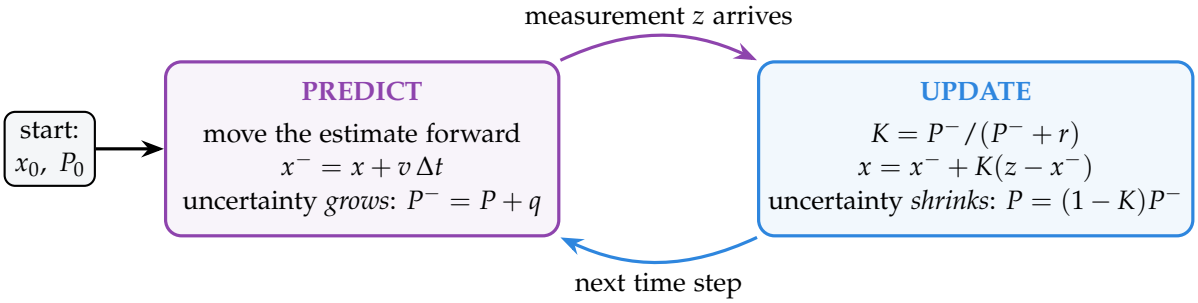


Figure 7: The Kalman filter alternates between two steps. Prediction widens the bell curve; measurement narrows it again.

From now on we use the standard short notation:

Symbol	Meaning
x, P	estimate and its variance *after* an update
x^-, P^-	estimate and variance *after* a prediction (before the measurement)
q	process noise: variance added at every prediction
r	measurement noise: variance of the sensor (here $r = 12$)
z	the measurement

5.1 The prediction step

The model moves the estimate forward according to how the system is expected to behave. If the car travels at 20 m/s, then 1 s later the predicted position is 20 m further down the road:

$$x_k^- = x_{k-1} + v\,\Delta t = x_{k-1} + 20.$$

At the same time the filter makes itself *less* sure, because the model is never perfect: the car might speed up a little, hit a bump, or change speed in a way the model did not include. So each prediction adds some extra variance called **process noise** q:

$$P_k^- = P_{k-1} + q.$$

In the video, $q = 1$ per second.

Why add the variances? Write the true motion as $x_k = x_{k-1} + v\Delta t + w_k$, where w_k is a random disturbance with variance q. The old error and the new disturbance are independent, so by fact (2) their variances add: $P^- = P + q$.

5.2 The update step

This is exactly Section 3:

$$K_k = \frac{P_k^-}{P_k^- + r}, \qquad x_k = x_k^- + K_k\,(z_k - x_k^-), \qquad P_k = (1 - K_k)\,P_k^-.$$

5.3 Watching the filter settle (full calculation)

The video starts the filter with a terrible first guess and variance $P_1^- = 100$: the filter admits it knows almost nothing. Let us do the first three steps by hand with $q = 1$ and $r = 12$.

Step-by-step: how the gain settles

Step 1. $P_1^- = 100$.

$$K_1 = \frac{100}{100 + 12} = \frac{100}{112} \approx 0.893 \;(\approx \tfrac{9}{10}), \qquad P_1 = (1 - 0.893)\cdot 100 = \frac{12}{112}\cdot 100 \approx 10.71.$$

The filter trusts the GPS almost completely, which is right: the model is still very uncertain.

Step 2. Predict: $P_2^- = 10.71 + 1 = 11.71$.

$$K_2 = \frac{11.71}{11.71 + 12} = \frac{11.71}{23.71} \approx 0.494 \;(\approx \tfrac{1}{2}), \qquad P_2 = \frac{12}{23.71}\cdot 11.71 \approx 5.93.$$

Step 3. Predict: $P_3^- = 5.93 + 1 = 6.93$.

$$K_3 = \frac{6.93}{18.93} \approx 0.366 \text{ (a little over } \tfrac{1}{3}\text{)}, \qquad P_3 = \frac{12}{18.93} \cdot 6.93 \approx 4.39.$$

Notice a handy shortcut: $P_k = (1 - K_k)P_k^- = \dfrac{r\,P_k^-}{P_k^- + r}$.

Continuing the same way gives the full table:

step k	P_k^- (after predict)	gain K_k	P_k (after update)
1	100.000	0.893	10.714
2	11.714	0.494	5.928
3	6.928	0.366	4.392
4	5.392	0.310	3.720
5	4.720	0.282	3.388
6	4.388	0.268	3.213
7	4.213	0.260	3.118
8	4.118	0.256	3.066
9	4.066	0.253	3.037
10	4.037	0.252	3.021
∞	**4**	**0.25**	**3**

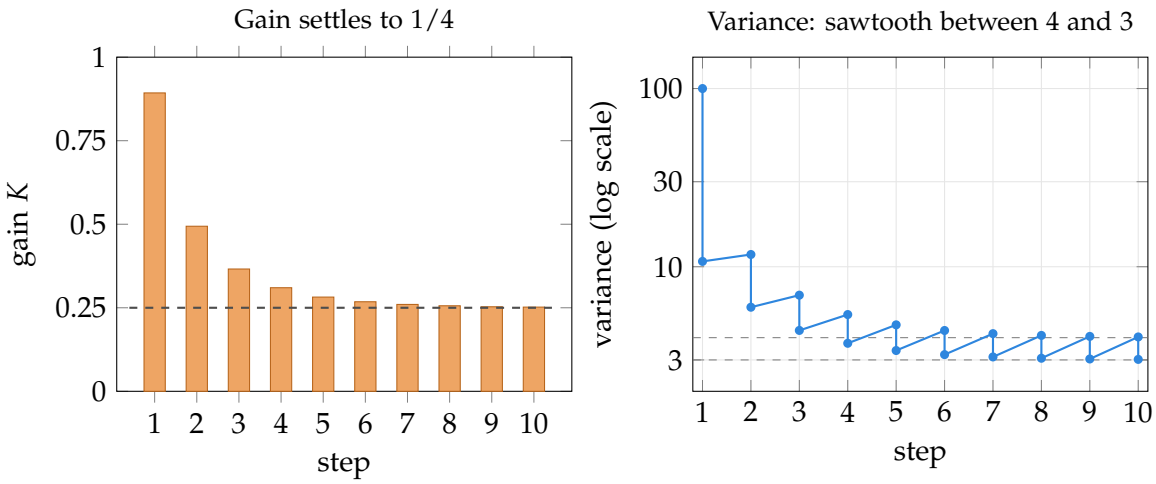


Figure 8: Left: the gain starts near 0.9 and settles near 0.25. Right: every prediction adds 1 (up), every update cuts the variance (down). In the end it moves between 4 and 3.

5.4 The steady state, calculated exactly

In the steady state, the variance after an update stops changing: $P_k = P_{k-1} = P$. Combining both steps:

$$P = \frac{r\,(P + q)}{(P + q) + r}.$$

With $q = 1$, $r = 12$:

$$\begin{aligned} P(P+13) &= 12(P+1) \\ P^2 + 13P &= 12P + 12 \\ P^2 + P - 12 &= 0 \\ (P+4)(P-3) = 0 \quad &\Longrightarrow \quad P = 3 \quad \text{(a variance cannot be negative)}. \end{aligned}$$

So $P^- = P + q = 4$ and $K = \dfrac{4}{4+12} = \dfrac{1}{4}$. This matches the video's formula

$$K = \frac{3+1}{3+1+12} = \frac{1}{4}.$$

In general, solving $P^2 + qP - qr = 0$ gives

$$P_\infty = \frac{-q + \sqrt{q^2 + 4qr}}{2}.$$

Check: $\dfrac{-1+\sqrt{1+48}}{2} = \dfrac{-1+7}{2} = 3$. ✓

The example was not chosen by hand

The first example (prediction variance 4, gain 1/4, new variance 3) was simply the filter *after* it had settled. Nobody told the filter to "trust the GPS by one quarter". That weight came out of the process noise ($q = 1$) and the sensor noise ($r = 12$) alone. When the filter's own prediction is uncertain it leans on the measurement; once it has built confidence in the model, it relies more on the prediction.

6 More than one number: position and velocity

Real systems usually have several quantities that change together, and some of them are never measured directly. Take the same car, but now estimate both **position** p and **velocity** v, while the GPS still reports only position.

6.1 The state vector and the covariance matrix

The state is now a vector:

$$\mathbf{x} = \begin{bmatrix} p \\ v \end{bmatrix}.$$

One variance is no longer enough. We use a 2×2 **covariance matrix**:

$$P = \begin{bmatrix} \mathrm{Var}(p) & \mathrm{Cov}(p,v) \\ \mathrm{Cov}(v,p) & \mathrm{Var}(v) \end{bmatrix}.$$

- The **diagonal** entries are the uncertainties of position and velocity on their own.
- The **off-diagonal** entries tell us how the two errors are *linked*. A positive value means: when the velocity estimate is too high, the position estimate tends to be too high as well.

Geometrically, the uncertainty is no longer a bell curve on a line but an *ellipse* in the position–velocity plane. A diagonal matrix gives an upright ellipse; off-diagonal entries tilt it.

6.2 The matrices of the model

For a time step $\Delta t = 1$ s:

$$F = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad H = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad R = 12.$$

- F (**state transition**) says: new position = old position + Δt·velocity; velocity stays the same.
- H (**measurement matrix**) says: the GPS sees the position only, not the velocity.
- Q (**process noise**) adds 1 to the position variance at each step, as in the 1-D case.[1]
- R is the GPS variance.

6.3 The matrix equations

These are the equations shown in the video. They have exactly the same structure as the 1-D ones.

The Kalman filter (matrix form)

Predict	$\hat{\mathbf{x}}^- = F\hat{\mathbf{x}}$	move the state forward
	$P^- = FPF^\top + Q$	move and grow the uncertainty
Update	$K = P^- H^\top \left(HP^- H^\top + R\right)^{-1}$	Kalman gain
	$\hat{\mathbf{x}} = \hat{\mathbf{x}}^- + K\left(\mathbf{z} - H\hat{\mathbf{x}}^-\right)$	correct with the surprise
	$P = (I - KH)\, P^-$	shrink the uncertainty

The 1-D filter is a special case. With one state, $F = 1$, $H = 1$, $Q = q$, $R = r$, the equations become $P^- = P + q$, $K = P^-/(P^- + r)$, $x = x^- + K(z - x^-)$, $P = (1 - K)P^-$: exactly Section 5.

6.4 Why $FPF^\top$? (Gaussian through a linear map)

If the error vector $\mathbf{e}$ has covariance P, then the error after the map, $F\mathbf{e}$, has covariance

$$\mathrm{Cov}(F\mathbf{e}) = \mathbb{E}\left[(F\mathbf{e})(F\mathbf{e})^\top\right] = F\,\mathbb{E}[\mathbf{e}\mathbf{e}^\top]\,F^\top = FPF^\top.$$

This is the matrix version of fact (1): "multiply by $a \Rightarrow$ variance times a^2" becomes "multiply by $F \Rightarrow$ covariance becomes $FPF^\top$". Then we add the independent process noise Q (matrix version of fact (2)).

[1] The video does not print Q and R for this part, but the numbers shown on screen are reproduced exactly by $Q = \mathrm{diag}(1, 0)$ and $R = 12$, the same values as in the 1-D example. A more physical choice of Q (random acceleration) is discussed in Section 8.

6.5 The worked 2-D example from the video

At the start of this example the filter believes:

$$\hat{\mathbf{x}} = \begin{bmatrix} 242.8 \\ 20 \end{bmatrix}, \qquad P = \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}.$$

Position and velocity errors are not linked yet (off-diagonals are 0), so the ellipse is upright.

Predict step (full matrix arithmetic)

State:

$$\hat{\mathbf{x}}^- = F\hat{\mathbf{x}} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 242.8 \\ 20 \end{bmatrix} = \begin{bmatrix} 242.8 + 20 \\ 20 \end{bmatrix} = \begin{bmatrix} 262.8 \\ 20 \end{bmatrix}.$$

Covariance, first FP:

$$FP = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix} = \begin{bmatrix} 1\cdot 3 + 1\cdot 0 & 1\cdot 0 + 1\cdot 4 \\ 0\cdot 3 + 1\cdot 0 & 0\cdot 0 + 1\cdot 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 0 & 4 \end{bmatrix}.$$

Then multiply by $F^\top = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$:

$$FPF^\top = \begin{bmatrix} 3 & 4 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3+4 & 0+4 \\ 0+4 & 0+4 \end{bmatrix} = \begin{bmatrix} 7 & 4 \\ 4 & 4 \end{bmatrix}.$$

Add the process noise:

$$P^- = \begin{bmatrix} 7 & 4 \\ 4 & 4 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 4 \\ 4 & 4 \end{bmatrix}.$$

Check by hand without matrices. New position $= p + v$, so $\mathrm{Var}(p+v) = \mathrm{Var}(p) + \mathrm{Var}(v) + 2\,\mathrm{Cov}(p,v) = 3 + 4 + 0 = 7$, plus process noise 1 gives 8. And $\mathrm{Cov}(p+v, v) = \mathrm{Cov}(p,v) + \mathrm{Var}(v) = 0 + 4 = 4$. ✓

In plain words

Position moves according to velocity. So if our velocity estimate is too high, the predicted position also ends up too far ahead. The errors become **linked**, and the off-diagonal entry 4 records this link. The correlation is $\rho = 4/\sqrt{8 \cdot 4} \approx 0.71$: a strong positive link, which is why the ellipse tilts.

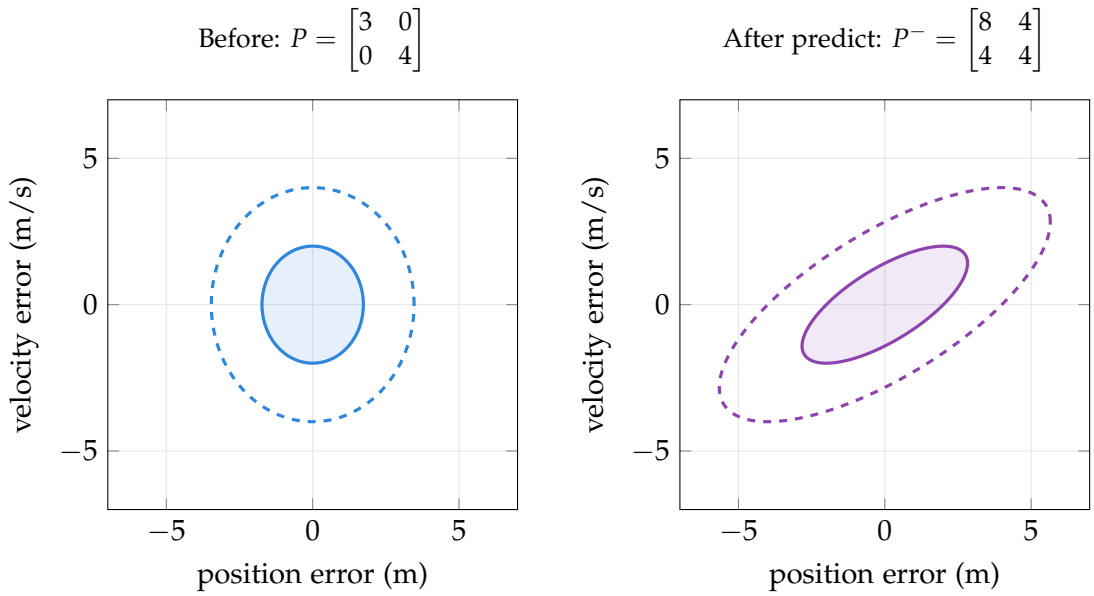


Figure 9: Uncertainty ellipses (1σ solid, 2σ dashed). Prediction stretches the ellipse along the position axis and tilts it: position and velocity errors are now linked.

Update step (full matrix arithmetic)

A GPS reading arrives. The numbers in the video correspond to a reading of $z = 252.8$ m, i.e. the car is 10 m *behind* where the model expected.

1. Innovation (surprise):

$$y = z - H\hat{\mathbf{x}}^- = 252.8 - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 262.8 \\ 20 \end{bmatrix} = 252.8 - 262.8 = -10.$$

2. Innovation variance $S = HP^-H^\top + R$. Since $H = [1 \;\; 0]$, $HP^-H^\top$ just picks the top-left entry:

$$S = 8 + 12 = 20.$$

3. Kalman gain. $P^-H^\top$ picks the first column of P^-:

$$P^-H^\top = \begin{bmatrix} 8 \\ 4 \end{bmatrix}, \qquad K = \frac{1}{20} \begin{bmatrix} 8 \\ 4 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.2 \end{bmatrix}.$$

4. New state:

$$\hat{\mathbf{x}} = \begin{bmatrix} 262.8 \\ 20 \end{bmatrix} + \begin{bmatrix} 0.4 \\ 0.2 \end{bmatrix} (-10) = \begin{bmatrix} 262.8 - 4 \\ 20 - 2 \end{bmatrix} = \begin{bmatrix} 258.8 \\ 18 \end{bmatrix}.$$

5. New covariance:

$$KH = \begin{bmatrix} 0.4 \\ 0.2 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 0.4 & 0 \\ 0.2 & 0 \end{bmatrix}, \qquad I - KH = \begin{bmatrix} 0.6 & 0 \\ -0.2 & 1 \end{bmatrix},$$

$$P = (I - KH)P^- = \begin{bmatrix} 0.6 & 0 \\ -0.2 & 1 \end{bmatrix} \begin{bmatrix} 8 & 4 \\ 4 & 4 \end{bmatrix} = \begin{bmatrix} 0.6 \cdot 8 & 0.6 \cdot 4 \\ -0.2 \cdot 8 + 4 & -0.2 \cdot 4 + 4 \end{bmatrix} = \begin{bmatrix} 4.8 & 2.4 \\ 2.4 & 3.2 \end{bmatrix}.$$

These are exactly the values shown in the video (position 258.8 m, velocity 18 m/s, covariance $\begin{bmatrix} 4.8 & 2.4 \\ 2.4 & 3.2 \end{bmatrix}$).

The GPS never measured velocity, yet velocity was corrected

The measurement said the car was *behind* the prediction. Because the filter knows that position and velocity errors are linked (off-diagonal 4), it reasons: "if I was too far ahead, I was probably also going too fast". So it lowered the velocity from 20 to 18 m/s. The second entry of the gain, $K_2 = \mathrm{Cov}(p, v)/S = 4/20 = 0.2$, is exactly this link turned into a correction. Notice also that the velocity variance dropped from 4 to 3.2.

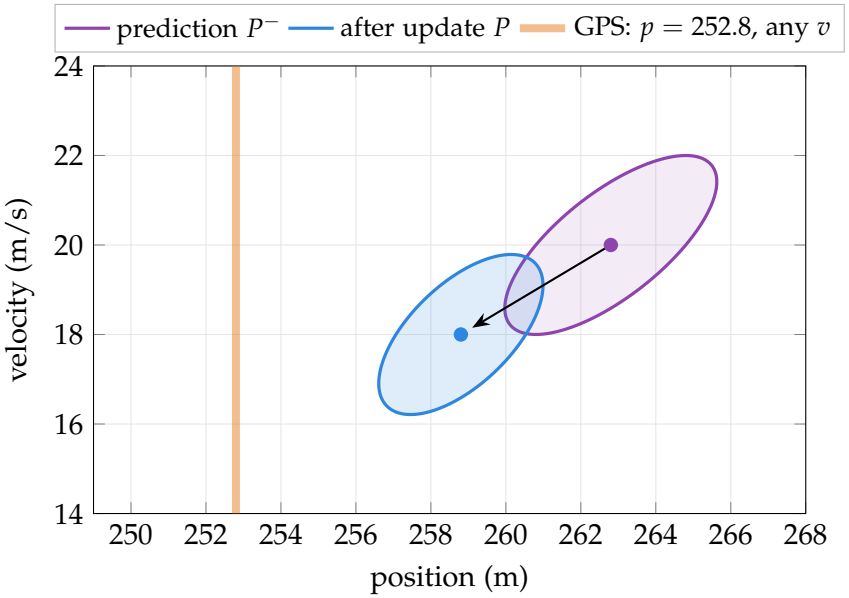


Figure 10: The GPS only constrains position (orange vertical band). Because the purple ellipse is tilted, pulling the position left also pulls the velocity down. The blue ellipse is smaller: we are more certain about *both* quantities.

6.6 One more cycle (our own extension)

Let us continue past the video to practise.

Cycle 2: predict

$$\hat{\mathbf{x}}^- = F\begin{bmatrix}258.8\\18\end{bmatrix} = \begin{bmatrix}258.8+18\\18\end{bmatrix} = \begin{bmatrix}276.8\\18\end{bmatrix} \quad \text{(this matches the video's next frame: 276.8 m).}$$

$$FP = \begin{bmatrix}1&1\\0&1\end{bmatrix}\begin{bmatrix}4.8&2.4\\2.4&3.2\end{bmatrix} = \begin{bmatrix}7.2&5.6\\2.4&3.2\end{bmatrix}, \qquad FPF^\top = \begin{bmatrix}7.2&5.6\\2.4&3.2\end{bmatrix}\begin{bmatrix}1&0\\1&1\end{bmatrix} = \begin{bmatrix}12.8&5.6\\5.6&3.2\end{bmatrix},$$

$$P^- = \begin{bmatrix}12.8&5.6\\5.6&3.2\end{bmatrix} + \begin{bmatrix}1&0\\0&0\end{bmatrix} = \begin{bmatrix}13.8&5.6\\5.6&3.2\end{bmatrix}.$$

Cycle 2: update with an invented reading $z = 280$ m

$$y = 280 - 276.8 = 3.2, \qquad S = 13.8 + 12 = 25.8, \qquad K = \frac{1}{25.8}\begin{bmatrix}13.8\\5.6\end{bmatrix} \approx \begin{bmatrix}0.535\\0.217\end{bmatrix}.$$

$$\hat{\mathbf{x}} = \begin{bmatrix} 276.8 \\ 18 \end{bmatrix} + \begin{bmatrix} 0.535 \\ 0.217 \end{bmatrix} (3.2) \approx \begin{bmatrix} 276.8 + 1.712 \\ 18 + 0.695 \end{bmatrix} = \begin{bmatrix} 278.51 \\ 18.69 \end{bmatrix}.$$

Using $P = P^- - K\,(HP^-)$ with $HP^- = [\,13.8 \;\; 5.6\,]$:

$$P = \begin{bmatrix} 13.8 & 5.6 \\ 5.6 & 3.2 \end{bmatrix} - \frac{1}{25.8} \begin{bmatrix} 13.8^2 & 13.8 \cdot 5.6 \\ 5.6 \cdot 13.8 & 5.6^2 \end{bmatrix} = \begin{bmatrix} 13.8 - 7.381 & 5.6 - 2.995 \\ 5.6 - 2.995 & 3.2 - 1.216 \end{bmatrix} \approx \begin{bmatrix} 6.42 & 2.60 \\ 2.60 & 1.98 \end{bmatrix}.$$

Observations. (1) The position gain is now 0.535, much larger than 0.25, because velocity uncertainty fed extra uncertainty into the predicted position (13.8 instead of 8). (2) The car was ahead of the prediction, so the filter *raised* the velocity estimate. (3) The velocity variance keeps falling ($4 \to 3.2 \to 1.98$): the filter is *learning the speed* from position readings alone.

6.7 Real-world versions

- **Apollo** tracked position and velocity in three dimensions (a 6-number state x, y, z, v_x, v_y, v_z), using star sightings that revealed only some parts of that state.
- **Phones** combine slow GPS updates with much faster accelerometer and gyroscope readings.
- **Electric cars** estimate the battery's state of charge (e.g. 64%), which cannot be measured directly, from voltage and current.

7 Derivation of the matrix Kalman gain (optional)

This section shows that the matrix formula for K comes from the same idea as in 1-D: minimise the leftover uncertainty.

Setup. The measurement is $\mathbf{z} = H\mathbf{x} + \mathbf{v}$ with noise covariance R. Let $\mathbf{e}^- = \mathbf{x} - \hat{\mathbf{x}}^-$ be the prediction error (covariance P^-), independent of $\mathbf{v}$.

Error after the update.

$$\mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}^- - K(H\mathbf{x} + \mathbf{v} - H\hat{\mathbf{x}}^-) = (I - KH)\,\mathbf{e}^- - K\mathbf{v}.$$

Its covariance (the two parts are independent, so their covariances add):

$$P = (I - KH)\,P^-\,(I - KH)^\top + KRK^\top.$$

This is called the *Joseph form*. It is the matrix version of $(1-K)^2\sigma_p^2 + K^2\sigma_g^2$. Expanding, with $S = HP^-H^\top + R$:

$$P = P^- - KHP^- - P^-H^\top K^\top + KSK^\top.$$

Minimise the total variance $\operatorname{tr}(P)$ (the sum of the diagonal = sum of all individual variances). Using the matrix-calculus rules $\frac{\partial}{\partial K}\operatorname{tr}(KA) = A^\top$ and $\frac{\partial}{\partial K}\operatorname{tr}(KSK^\top) = 2KS$ (for symmetric S):

$$\frac{\partial \operatorname{tr}(P)}{\partial K} = -(HP^-)^\top - P^-H^\top + 2KS = -2P^-H^\top + 2KS = 0 \quad \Longrightarrow \quad K = P^-H^\top S^{-1}.$$

Simplify P**.** Substituting $KS = P^-H^\top$ gives $KSK^\top = P^-H^\top K^\top$, which cancels the third term:

$$P = P^- - KHP^- = (I - KH)\,P^-.$$

In 1-D ($H = 1$) this is exactly $K = \sigma_p^2/(\sigma_p^2 + \sigma_g^2)$ and $P = (1-K)\sigma_p^2$.

8 Limits and variations

8.1 Assumption 1: the system is linear

The Kalman filter assumes the prediction and the measurement are *straight-line* (linear) relationships. This matters because a bell curve that passes through a linear function stays a bell curve. If $X \sim \mathcal{N}(\mu, \sigma^2)$ and $Y = aX + b$, then

$$Y \sim \mathcal{N}(a\mu + b,\ a^2\sigma^2).$$

A curved (nonlinear) function bends the bell curve into a lopsided shape, and the simple formulas no longer hold.

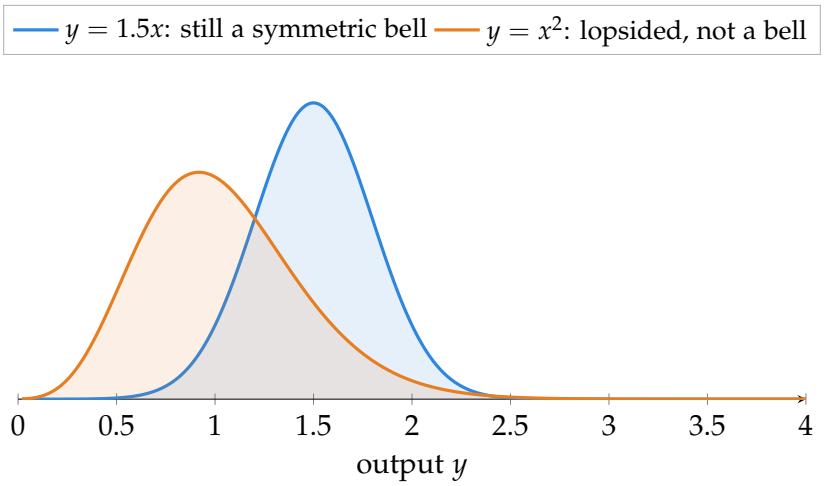


Figure 11: The same input bell curve (x with mean 1, standard deviation 0.2) sent through a linear and a nonlinear function.

Real systems are not always linear: a turning vehicle, or a radar that measures range and angle, involves curved relationships. Common solutions:

- **Extended Kalman Filter (EKF):** approximates the nonlinear system by a straight line (its tangent, via derivatives/Jacobians) around the current estimate.
- **Unscented Kalman Filter (UKF):** sends a small, carefully chosen set of "sigma points" through the true nonlinear function and rebuilds a bell curve from where they land.
- **Particle filter:** when one bell curve cannot describe the uncertainty at all (e.g. two possible locations), it uses a large cloud of samples instead.

8.2 Assumption 2: the noise levels are right

The filter can also fail by being given the wrong noise levels. Suppose we tell it the model is perfect: $q = 0$. Then the prediction never adds uncertainty, and by the precision rule every update adds $1/r$ of precision:

$$\frac{1}{P_n} = \frac{1}{P_0} + \frac{n}{r}.$$

With $P_0 = 100$ and $r = 12$, after $n = 100$ steps: $1/P_{100} = 0.01 + 8.33 = 8.34$, so $P_{100} \approx 0.12$ and

$$K \approx \frac{0.12}{0.12 + 12} \approx 0.01.$$

The filter now accepts only 1% of each surprise. If the car really changes speed, the estimate drifts away from reality while the filter becomes *more and more confident* that it is right. This is why choosing realistic Q and R is a central part of using a Kalman filter in practice.

In plain words

Too small Q: filter is overconfident, ignores the sensor, lags behind real changes.
Too large Q: filter trusts every noisy reading, estimate becomes jittery.
R is usually found from the sensor's datasheet or by recording readings of something that is not moving.

A common, more physical choice for the position–velocity model is to assume random accelerations with variance σ_a^2, which gives

$$Q = \sigma_a^2 \begin{bmatrix} \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 \\ \frac{1}{2}\Delta t^3 & \Delta t^2 \end{bmatrix}.$$

9 Practice problems with full solutions

Problem 1 (Thermometer). *A model predicts that a room is at 20°C with variance 2. A thermometer reads 23°C with variance 6. Find the gain, the new estimate and the new variance. Check the variance with the precision rule.*

Solution

$$K = \frac{2}{2+6} = \frac{1}{4}, \qquad y = 23 - 20 = 3, \qquad \hat{x} = 20 + \tfrac{1}{4} \cdot 3 = 20.75^\circ\text{C},$$

$$\sigma^2_{\text{new}} = \left(1 - \tfrac{1}{4}\right) \cdot 2 = 1.5.$$

Check: $\frac{1}{2} + \frac{1}{6} = \frac{3}{6} + \frac{1}{6} = \frac{4}{6} = \frac{2}{3}$, so $\sigma^2 = \frac{3}{2} = 1.5$. ✓

Problem 2 (Equal trust). *Prediction 50 m with variance 9; measurement 56 m with variance 9. What happens?*

Solution

$K = 9/18 = 1/2$, so $\hat{x} = 50 + \frac{1}{2} \cdot 6 = 53$ m, the plain average. Variance $= \frac{1}{2} \cdot 9 = 4.5$: two equally good sources halve the variance. A 50/50 average is correct *only* in this special case.

Problem 3 (Two full cycles by hand). *A car starts at $x_0 = 0$ m with variance $P_0 = 4$. It moves at 10 m/s, $\Delta t = 1$ s, process noise $q = 1$, GPS variance $r = 4$. The GPS reads $z_1 = 12$ m, then $z_2 = 20$ m. Run two predict–update cycles.*

Solution

Cycle 1.

$$\text{Predict:} \quad x_1^- = 0 + 10 = 10, \qquad P_1^- = 4 + 1 = 5$$

$$\text{Update:} \quad K_1 = \frac{5}{5+4} = \frac{5}{9} \approx 0.556, \qquad y_1 = 12 - 10 = 2,$$

$$x_1 = 10 + \frac{5}{9} \cdot 2 = \frac{100}{9} \approx 11.111 \text{ m}, \qquad P_1 = \frac{4}{9} \cdot 5 = \frac{20}{9} \approx 2.222$$

Cycle 2.

$$\text{Predict:} \quad x_2^- = \frac{100}{9} + 10 = \frac{190}{9} \approx 21.111, \qquad P_2^- = \frac{20}{9} + 1 = \frac{29}{9} \approx 3.222$$

$$\text{Update:} \quad K_2 = \frac{29/9}{29/9 + 36/9} = \frac{29}{65} \approx 0.446, \qquad y_2 = 20 - \frac{190}{9} = -\frac{10}{9} \approx -1.111,$$

$$x_2 = \frac{190}{9} - \frac{29}{65} \cdot \frac{10}{9} = \frac{268}{13} \approx 20.615 \text{ m}, \qquad P_2 = \frac{36}{65} \cdot \frac{29}{9} = \frac{116}{65} \approx 1.785.$$

The gain falls ($0.556 \to 0.446$) as the filter becomes more confident.

Problem 4 (Steady state). *For a 1-D filter with $q = 4$ and $r = 8$, find the steady-state variances P, P^- and the gain K.*

Solution

$$P_\infty = \frac{-q + \sqrt{q^2 + 4qr}}{2} = \frac{-4 + \sqrt{16 + 128}}{2} = \frac{-4 + 12}{2} = 4,$$

$$P^- = 4 + 4 = 8, \qquad K = \frac{8}{8 + 8} = \frac{1}{2}.$$

Check: $P = \frac{rP^-}{P^- + r} = \frac{8 \cdot 8}{16} = 4$. ✓ (More process noise than in the video, so the filter trusts the GPS more: $K = \frac{1}{2}$ instead of $\frac{1}{4}$.)

Problem 5 (2-D update). *After a prediction,* $\hat{\mathbf{x}}^- = \begin{bmatrix} 100 \\ 15 \end{bmatrix}$, $P^- = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$, $H = [1 \;\; 0]$, $R = 10$. *The GPS reads* $z = 104$. *Find* K, $\hat{\mathbf{x}}$ *and* P.

Solution

$S = 10 + 10 = 20$, $K = \frac{1}{20} \begin{bmatrix} 10 \\ 5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}$, $y = 104 - 100 = 4$.

$$\hat{\mathbf{x}} = \begin{bmatrix} 100 + 0.5 \cdot 4 \\ 15 + 0.25 \cdot 4 \end{bmatrix} = \begin{bmatrix} 102 \\ 16 \end{bmatrix}, \qquad P = P^- - K \begin{bmatrix} 10 & 5 \end{bmatrix} = \begin{bmatrix} 10 - 5 & 5 - 2.5 \\ 5 - 2.5 & 5 - 1.25 \end{bmatrix} = \begin{bmatrix} 5 & 2.5 \\ 2.5 & 3.75 \end{bmatrix}.$$

The car was ahead of the prediction, so the velocity estimate went *up* by 1 m/s, even though velocity was never measured.

Problem 6 (Concept check). *(a) What is* K *if* $R = 0$*? (b) If* $P^- = 0$*? (c) Why can the new variance never be larger than the predicted one?*

Solution

(a) $K = 1$: perfect sensor, copy the measurement. (b) $K = 0$: perfect prediction, ignore the measurement. (c) Precisions add: $1/\sigma^2_{\text{new}} = 1/\sigma_p^2 + 1/\sigma_g^2 \geq 1/\sigma_p^2$, so $\sigma^2_{\text{new}} \leq \sigma_p^2$.

10 Summary cheat sheet

Everything on one page

1-D filter (state x, variance P, process noise q, sensor noise r):

$$\text{predict:}\quad x^- = x + v\Delta t, \qquad P^- = P + q$$

$$\text{update:}\quad K = \frac{P^-}{P^- + r}, \qquad x = x^- + K(z - x^-), \qquad P = (1 - K)P^-$$

Matrix filter:

$$\text{predict:}\quad \hat{\mathbf{x}}^- = F\hat{\mathbf{x}}, \qquad P^- = FPF^\top + Q$$

$$\text{update:}\quad K = P^- H^\top (HP^- H^\top + R)^{-1}, \qquad \hat{\mathbf{x}} = \hat{\mathbf{x}}^- + K(\mathbf{z} - H\hat{\mathbf{x}}^-), \qquad P = (I - KH)P^-$$

Useful facts:

- Precisions add: $\dfrac{1}{\sigma^2_{\text{new}}} = \dfrac{1}{\sigma^2_p} + \dfrac{1}{\sigma^2_g}$.
- The gain minimises the leftover variance $(1-K)^2\sigma_p^2 + K^2\sigma_g^2$.
- 1-D steady state: $P_\infty = \frac{1}{2}\big(-q + \sqrt{q^2 + 4qr}\big)$.
- Prediction widens the bell curve; measurement narrows it.
- Off-diagonal covariance lets a measurement of one quantity correct another (hidden) one.

Video numbers: 100 & 106 $\to$ 101.5; variances 4 & 12 $\to$ 3; $K = \frac{1}{4}$; 2-D: $(242.8, 20) \to (262.8, 20) \to (258.8, 18) \to (276.8, 18)$.

A Python code for the examples

This short script reproduces the numbers in Sections 3, 5 and 6.

```
import numpy as np

# ---- 1-D single update (Section 3)
xp, Pp, z, r = 100.0, 4.0, 106.0, 12.0
K = Pp / (Pp + r)
x = xp + K * (z - xp); P = (1 - K) * Pp
print(K, x, P) # 0.25 101.5 3.0

# ---- 1-D gain settling (Section 5)
P, q = 100.0, 1.0
for k in range(1, 11):
   Pm = P if k == 1 else P + q # predict (first step starts at 100)
   K = Pm / (Pm + r); P = (1 - K) * Pm
   print(k, round(Pm, 3), round(K, 3), round(P, 3))

# ---- 2-D example (Section 6)
F = np.array([[1, 1], [0, 1.]]); H = np.array([[1, 0.]])
Q = np.diag([1., 0.]); R = np.array([[12.]])
x = np.array([242.8, 20.]); P = np.diag([3., 4.])

def predict(x, P):
   return F @ x, F @ P @ F.T + Q

def update(x, P, z):
   S = H @ P @ H.T + R
   K = P @ H.T @ np.linalg.inv(S)
   x = x + (K @ (np.atleast_1d(z) - H @ x))
   P = (np.eye(2) - K @ H) @ P
   return x, P

x, P = predict(x, P); print(x, P) # [262.8 20] [[8 4] [4 4]]
x, P = update(x, P, 252.8); print(x, P) # [258.8 18] [[4.8 2.4] [2.4 3.2]]
```