Streaming solutions to least-squares problems

In our discussion of least-squares so far, we have focussed on static problems: a set of measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{e}$ comes in all at once, and we use them all to estimate $\boldsymbol{x}_0$.

In this section, we will shift our focus to **streaming problems**. We observe

$$\begin{aligned} \boldsymbol{y}_0 &= \boldsymbol{A}_0\boldsymbol{x}_1 + \boldsymbol{e}_0 \\ \boldsymbol{y}_1 &= \boldsymbol{A}_1\boldsymbol{x}_2 + \boldsymbol{e}_1 \\ &\vdots \\ \boldsymbol{y}_k &= \boldsymbol{A}_k\boldsymbol{x}_k + \boldsymbol{e}_k \\ &\vdots \end{aligned}$$

At each time k, we want to form the best estimate of $\boldsymbol{x}_k$ from the observations $\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ seen up to that point exploiting some known structure regarding how the $\boldsymbol{x}_k$ are related to each other. Moreover, we would like to do this in an efficient manner. The size of the problem is growing with k — rather than resolving the problem from scratch every time, we would like a principled (and fast) way to **update** the solution when a new observation is made.

We will consider two basic frameworks:

1. Recursive Least Squares (RLS):
   The vector $\boldsymbol{x}_k = \boldsymbol{x}_*$ for all k, i.e. we are estimating a static vector $\boldsymbol{x}_*$.
2. The Kalman filter:
   The vector $\boldsymbol{x}_k$ moves at every times step, and we have a (linear) dynamical model for how it moves.

In both of these frameworks, the measurements matrices $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots$ can be different, and can even have a different number of rows. We will assume that the total number of measurements we have seen at any point exceeds the number of unknowns, and if we form

$$\underline{\boldsymbol{A}}_k = \begin{bmatrix} \boldsymbol{A}_0 \\ \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_k \end{bmatrix}$$

then $\underline{\boldsymbol{A}}_k^{\mathrm{T}} \underline{\boldsymbol{A}}_k$ is invertible. Generalizing what we say to rank-deficient systems is not hard, but this assumption makes the discussion easier.

The key piece of mathematical technology we need is the **matrix inversion lemma**.

The Matrix Inversion Lemma

The matrix inversion lemma shows us how the solution to a system of equations can be efficiently updated. Here we state a slightly simplified version of the result: If $\boldsymbol{W}$ is an $N \times N$ invertible matrix and $\boldsymbol{X}$ is an $R \times N$ matrix, then the following identity holds:

$$(\boldsymbol{W} + \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1} = \boldsymbol{W}^{-1} - \boldsymbol{W}^{-1}\boldsymbol{X}^{\mathrm{T}}(\mathbf{I} + \boldsymbol{X}\boldsymbol{W}^{-1}\boldsymbol{X}^{\mathrm{T}})^{-1}\boldsymbol{X}\boldsymbol{W}^{-1}$$

This is a special case of the *Sherman-Morrison-Woodbury* identity, and is straightforward to prove (see the Technical Details at the end of these notes). The point is that if $\boldsymbol{W}^{-1}$ has already been calculated, then finding a solution to $(\boldsymbol{W} + \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})\boldsymbol{w} = \boldsymbol{v}$ costs $O(N^2R) + O(NR^2) + O(R^3)$ instead of $O(N^3)$. If R is very small compared to N, this can be a significant savings.

Updating least-squares solutions

We can apply this fact to efficiently update the solution to least-squares problems as new measurements become available.

Suppose we have observed

$$\boldsymbol{y}_0 = \boldsymbol{A}_0 \boldsymbol{x}_* + \boldsymbol{e}_0$$

and have formed the least-squares estimate

$$\widehat{\boldsymbol{x}}_0 = (\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{A}_0)^{-1} \boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{y}_0.$$

Now we observe

$$\boldsymbol{y}_1 = \boldsymbol{A}_1 \boldsymbol{x}_* + \boldsymbol{e}_1,$$

where $\boldsymbol{A}_1$ is an $M_1 \times N$ matrix with $M_1 \ll N$. Given $\boldsymbol{y}_0$ and $\boldsymbol{y}_1$, the full least-squares estimate is formed from the system of equations

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_0 \\ \boldsymbol{A}_1 \end{bmatrix} \boldsymbol{x}_* + \begin{bmatrix} \boldsymbol{e}_0 \\ \boldsymbol{e}_1 \end{bmatrix},$$

resulting in

$$\widehat{\boldsymbol{x}}_1 = \left(\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{A}_0 + \boldsymbol{A}_1^{\mathrm{T}} \boldsymbol{A}_1 \right)^{-1} (\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{y}_0 + \boldsymbol{A}_1^{\mathrm{T}} \boldsymbol{y}_1).$$

Now let $\boldsymbol{P}_k$ be the inverse of the aggregated system we would like to solve at each step:

$$\begin{aligned} \boldsymbol{P}_0 &= (\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{A}_0)^{-1} \\ \boldsymbol{P}_1 &= (\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{A}_0 + \boldsymbol{A}_1^{\mathrm{T}} \boldsymbol{A}_1)^{-1}. \end{aligned}$$

Then using the matrix inversion lemma with $\boldsymbol{W} = \boldsymbol{A}_0^T \boldsymbol{A}_0 = \boldsymbol{P}_0^{-1}$ and $\boldsymbol{X} = \boldsymbol{A}_1$ gives us the update

$$\boldsymbol{P}_1 = \boldsymbol{P}_0 - \boldsymbol{P}_0 \boldsymbol{A}_1^{\mathrm{T}} (\mathbf{I} + \boldsymbol{A}_1 \boldsymbol{P}_0 \boldsymbol{A}_1^{\mathrm{T}})^{-1} \boldsymbol{A}_1 \boldsymbol{P}_0.$$

When the number of new measurements (rows in $\boldsymbol{A}_1$) is small, then the system of equations $\mathbf{I} + \boldsymbol{A}_1 \boldsymbol{P}_0 \boldsymbol{A}_1^{\mathrm{T}}$ can be much easier to handle than $\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{A}_0 + \boldsymbol{A}_1^{\mathrm{T}} \boldsymbol{A}_1 = \boldsymbol{P}_1^{-1}$. For example, suppose we see just one new measurement, so the matrix $\boldsymbol{A}_1$ has just one row: $\boldsymbol{A}_1 = \boldsymbol{a}_1^{\mathrm{T}}$, $\boldsymbol{a}_1 \in \mathbb{R}^N$. Then

$$y_1 = \boldsymbol{a}_1^{\mathrm{T}} \boldsymbol{x}_* + e_1,$$

and

$$\widehat{\boldsymbol{x}}_1 = \left[\boldsymbol{P}_0 - \boldsymbol{P}_0 \boldsymbol{a}_1 (1 + \boldsymbol{a}_1^{\mathrm{T}} \boldsymbol{P}_0 \boldsymbol{a}_1)^{-1} \boldsymbol{a}_1^{\mathrm{T}} \boldsymbol{P}_0 \right] (\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{y}_0 + y_1 \boldsymbol{a}_1).$$

Set $\boldsymbol{u} = \boldsymbol{P}_0 \boldsymbol{a}_1$. Then

$$\begin{aligned} \widehat{\boldsymbol{x}}_1 &= \widehat{\boldsymbol{x}}_0 + y_1 \boldsymbol{u} - \frac{\boldsymbol{a}_1^{\mathrm{T}} \widehat{\boldsymbol{x}}_0}{1 + \boldsymbol{a}_1^{\mathrm{T}} \boldsymbol{u}} \boldsymbol{u} - \frac{y_1 \cdot \boldsymbol{a}_1^{\mathrm{T}} \boldsymbol{u}}{1 + \boldsymbol{a}_1^{\mathrm{T}} \boldsymbol{u}} \boldsymbol{u} \\ &= \widehat{\boldsymbol{x}}_0 + \left(\frac{1}{1 + \boldsymbol{a}_1^{\mathrm{T}} \boldsymbol{u}} \right) (y_1 - \boldsymbol{a}_1^{\mathrm{T}} \widehat{\boldsymbol{x}}_0) \, \boldsymbol{u}. \end{aligned}$$

Thus we can update the solution with one vector-matrix multiply (which has cost $O(N^2)$) and two inner products (with cost $O(N)$).

In addition, we can carry forward the "information matrix" using the update

$$\boldsymbol{P}_1 = \boldsymbol{P}_0 - \frac{1}{1 + \boldsymbol{a}_1^{\mathrm{T}} \boldsymbol{u}} \boldsymbol{u} \boldsymbol{u}^{\mathrm{T}}.$$

In general (for M_1 new measurements), we have

$$\begin{aligned} \widehat{\boldsymbol{x}}_1 &= \boldsymbol{P}_1 (\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{y}_0 + \boldsymbol{A}_1^{\mathrm{T}} \boldsymbol{y}_1) \\ &= \boldsymbol{P}_1 (\boldsymbol{P}_0^{-1} \widehat{\boldsymbol{x}}_0 + \boldsymbol{A}_1^{\mathrm{T}} \boldsymbol{y}_1), \end{aligned}$$

and since

$$\boldsymbol{P}_0^{-1} = \boldsymbol{P}_1^{-1} - \boldsymbol{A}_1^{\mathrm{T}} \boldsymbol{A}_1,$$

this implies

$$\begin{aligned}\widehat{\boldsymbol{x}}_1 &= \boldsymbol{P}_1\left(\boldsymbol{P}_1^{-1}\widehat{\boldsymbol{x}}_0 - \boldsymbol{A}_1^{\mathrm{T}}\boldsymbol{A}_1\widehat{\boldsymbol{x}}_0 + \boldsymbol{A}_1^{\mathrm{T}}\boldsymbol{y}_1\right)\\ &= \widehat{\boldsymbol{x}}_0 + \boldsymbol{K}_1(\boldsymbol{y}_1 - \boldsymbol{A}_1\widehat{\boldsymbol{x}}_0),\end{aligned}$$

where $\boldsymbol{K}_1$ is the "gain matrix"

$$\boldsymbol{K}_1 = \boldsymbol{P}_1\boldsymbol{A}_1^{\mathrm{T}}.$$

The update for $\boldsymbol{P}_1$ is

$$\begin{aligned}\boldsymbol{P}_1 &= \boldsymbol{P}_0 - \boldsymbol{P}_0\boldsymbol{A}_1^{\mathrm{T}}(\mathbf{I} + \boldsymbol{A}_1\boldsymbol{P}_0\boldsymbol{A}_1^{\mathrm{T}})^{-1}\boldsymbol{A}_1\boldsymbol{P}_0\\ &= \boldsymbol{P}_0 - \boldsymbol{U}(\mathbf{I} + \boldsymbol{A}_1\boldsymbol{U})^{-1}\boldsymbol{U}^{\mathrm{T}},\end{aligned}$$

where $\boldsymbol{U} = \boldsymbol{P}_0\boldsymbol{A}_1^{\mathrm{T}}$ is an $N \times M_1$ matrix, and $\mathbf{I} + \boldsymbol{A}_1\boldsymbol{U}$ is $M_1 \times M_1$. So the cost of the update is

- $O(M_1N^2)$ to compute $\boldsymbol{U} = \boldsymbol{P}_0\boldsymbol{A}_1^{\mathrm{T}}$,
- $O(M_1^2N)$ to compute $\boldsymbol{A}_1\boldsymbol{U}$,
- $O(M_1^3)$ to invert[1] $(\mathbf{I} + \boldsymbol{A}_1\boldsymbol{U})^{-1}$,
- $O(M_1^2N)$ to compute $(\mathbf{I} + \boldsymbol{A}_1\boldsymbol{U})^{-1}\boldsymbol{U}^{\mathrm{T}}$,
- $O(M_1N^2)$ to take the result of the last step and apply $\boldsymbol{U}$,
- $O(N^2)$ to subtract the result of the last step from $\boldsymbol{P}_0$.

So assuming that $M_1 < N$, the overall cost is $O(M_1N^2)$, which is on the order of M_1 vector-matrix multiplies.

[1] In practice, it is probably more stable to find and update a factorization of this matrix. But the cost is the same.

Recursive Least Squares (RLS)

Given

$$\begin{aligned}
\boldsymbol{y}_0 &= \boldsymbol{A}_0 \boldsymbol{x}_* + \boldsymbol{e}_0 \\
\boldsymbol{y}_1 &= \boldsymbol{A}_1 \boldsymbol{x}_* + \boldsymbol{e}_1 \\
&\vdots \\
\boldsymbol{y}_k &= \boldsymbol{A}_k \boldsymbol{x}_* + \boldsymbol{e}_k \\
&\vdots,
\end{aligned}$$

RLS is an **online algorithm** for computing the best estimate for $\boldsymbol{x}_*$ from all the measurements it has seen up to the current time.

Recursive Least Squares

Initialize: ($\boldsymbol{y}_0$ appears)

$$\boldsymbol{P}_0 = (\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{A}_0)^{-1}$$

$$\widehat{\boldsymbol{x}}_0 = \boldsymbol{P}_0 (\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{y}_0)$$

for $k = 1, 2, 3, \ldots$ **do**

($\boldsymbol{y}_k$ appears)

$$\boldsymbol{P}_k = \boldsymbol{P}_{k-1} - \boldsymbol{P}_{k-1} \boldsymbol{A}_k^{\mathrm{T}} (\mathbf{I} + \boldsymbol{A}_k \boldsymbol{P}_{k-1} \boldsymbol{A}_k^{\mathrm{T}})^{-1} \boldsymbol{A}_k \boldsymbol{P}_{k-1}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k \boldsymbol{A}_k^{\mathrm{T}}$$

$$\widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{x}}_{k-1} + \boldsymbol{K}_k (\boldsymbol{y}_k - \boldsymbol{A}_k \widehat{\boldsymbol{x}}_{k-1})$$

end for

The Kalman Filter

The RLS algorithm for updating the least squares estimate given a series of vector observations looked like a "filter": new data comes in, and we use it (along with collected knowledge of the old data) to produce a new output.

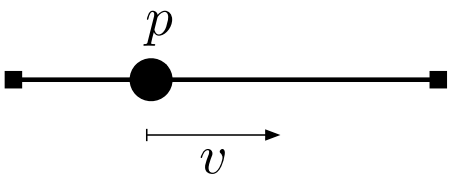
In the previous section, $\boldsymbol{x}_*$ was fixed for the entire sequence of observations. The **Kalman filter** incorporates **dynamics** for the unknown vector $\boldsymbol{x}$ into our estimation framework. It addresses the case where $\boldsymbol{x}$ **changes** from observation to observation in a manner which we can model.

The classic example is trying to estimate the position and velocity of an airplane. If the plane is in motion, these will of course change over time. But the path of the plane is somewhat predictable, and there are real physical constraints on how quickly it can turn and/or accelerate.

Our dynamical models will be linear. That is, we will assume that we can approximate the solution at the next time step $\boldsymbol{x}_{k+1}$ by applying a known $N \times N$ matrix $\boldsymbol{F}_k$ to the current solution:

$$\boldsymbol{x}_{k+1} \approx \boldsymbol{F}_k \boldsymbol{x}_k.$$

Here is a quick example of how this might work. Suppose that we are trying to estimate the position p and velocity v of a particle traveling in one dimension:



In this case, our unknowns are a two-vector:

$$\boldsymbol{x}_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix}.$$

If we expect the velocity to be almost the same from measurement to measurement, our dynamics equations would look like

$$\begin{aligned} p_{k+1} &= p_k + \alpha_k v_k \quad && (+ \text{ error}) \\ v_{k+1} &= v_k && (+ \text{ error}), \end{aligned}$$

where α_k would be proportional to the time elapsed between $k+1$ and k. In terms of the $\boldsymbol{x}_k$, we can write

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}_k \boldsymbol{x}_k, \quad \text{where} \quad \boldsymbol{F}_k = \begin{bmatrix} 1 & \alpha_k \\ 0 & 1 \end{bmatrix}.$$

Setting up the system

As before, we are making noisy observations of an unknown vector:

$$\boldsymbol{y}_k = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{e}_k, \quad \boldsymbol{e}_k = \text{“measurement error”}.$$

There can be a different number of measurements at each step, but all of the $\boldsymbol{x}_k$ have length N. We will say that $\boldsymbol{y}_k \in \mathbb{R}^{M_k}$ and the $\boldsymbol{A}_k$ are $M_k \times N$ matrices.

The $\boldsymbol{x}_k$ are dynamic; we model the transition from step k to $k+1$ using

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}_k \boldsymbol{x}_k + \boldsymbol{\epsilon}_k, \quad \boldsymbol{\epsilon}_k = \text{“state error”}.$$

The $\boldsymbol{F}_k$ are all $N \times N$.

It is possible to incorporate the correlation structure of the $\boldsymbol{e}_k$ and $\boldsymbol{\epsilon}_k$ into our estimates, but we will start with the standard least-squares framework.

Notation: Since the $\boldsymbol{x}_k$ are linked together through the dynamical model, the measurements $y_k = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{e}_k$ also give us indirect information about $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{k-1}$. As such, our estimate of **all** of $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k$ will change at time k. We will use the notation

$$\widehat{\boldsymbol{x}}_{\ell|k} = \text{least-squares estimate of } \boldsymbol{x}_\ell \text{ at time } k.$$

Step $k = 0$. We have observed

$$\boldsymbol{y}_0 = \boldsymbol{A}_0 \boldsymbol{x}_0 + \boldsymbol{e}_0.$$

We form the least-squares estimate

$$\widehat{\boldsymbol{x}}_{0|0} = (\boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{A}_0)^{-1} \boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{y}_0.$$

Step $k = 1$. We have the following system of equations

$$\begin{aligned} \boldsymbol{y}_0 &= \boldsymbol{A}_0 \boldsymbol{x}_0 + \boldsymbol{e}_0 \\ \boldsymbol{0} &= \boldsymbol{F}_0 \boldsymbol{x}_0 - \boldsymbol{x}_1 + \boldsymbol{\epsilon}_0 \\ \boldsymbol{y}_1 &= \boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{e}_1. \end{aligned}$$

We can write this compactly as $\underline{\boldsymbol{y}} = \underline{\boldsymbol{A}}_1 \underline{\boldsymbol{x}} + \underline{\boldsymbol{e}}_1$, where

$$\underbrace{\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{0} \\ \boldsymbol{y}_1 \end{bmatrix}}_{\underline{\boldsymbol{y}}_1} = \underbrace{\begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{0} \\ \boldsymbol{F}_0 & -\mathbf{I} \\ \boldsymbol{0} & \boldsymbol{A}_1 \end{bmatrix}}_{\underline{\boldsymbol{A}}_1} \underbrace{\begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \end{bmatrix}}_{\underline{\boldsymbol{x}}_1} + \underbrace{\begin{bmatrix} \boldsymbol{e}_0 \\ \boldsymbol{\epsilon}_0 \\ \boldsymbol{e}_1 \end{bmatrix}}_{\underline{\boldsymbol{e}}_1}.$$

This system is $(M_0 + N + M_1) \times 2N$; we can form the least-squares estimate using

$$\begin{bmatrix} \widehat{\boldsymbol{x}}_{0|1} \\ \widehat{\boldsymbol{x}}_{1|1} \end{bmatrix} = (\underline{\boldsymbol{A}}_1^{\mathrm{T}} \underline{\boldsymbol{A}}_1)^{-1} \underline{\boldsymbol{A}}_1^{\mathrm{T}} \underline{\boldsymbol{y}}_1.$$

Step $k = 2$. Now we have the following system of equations

$$\begin{aligned} \boldsymbol{y}_0 &= \boldsymbol{A}_0 \boldsymbol{x}_0 + \boldsymbol{e}_0 \\ \mathbf{0} &= \boldsymbol{F}_0 \boldsymbol{x}_0 - \boldsymbol{x}_1 + \boldsymbol{\epsilon}_0 \\ \boldsymbol{y}_1 &= \boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{e}_1 \\ \mathbf{0} &= \boldsymbol{F}_1 \boldsymbol{x}_1 - \boldsymbol{x}_2 + \boldsymbol{\epsilon}_1 \\ \boldsymbol{y}_2 &= \boldsymbol{A}_2 \boldsymbol{x}_2 + \boldsymbol{e}_2, \end{aligned}$$

which we can rewrite as

$$\underbrace{\begin{bmatrix} \boldsymbol{y}_0 \\ \mathbf{0} \\ \boldsymbol{y}_1 \\ \mathbf{0} \\ \boldsymbol{y}_2 \end{bmatrix}}_{\underline{\boldsymbol{y}}_2} = \underbrace{\begin{bmatrix} \boldsymbol{A}_0 & \mathbf{0} & \mathbf{0} \\ \boldsymbol{F}_0 & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{F}_1 & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{A}_2 \end{bmatrix}}_{\underline{\boldsymbol{A}}_2} \underbrace{\begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}}_{\underline{\boldsymbol{x}}_2} + \underbrace{\begin{bmatrix} \boldsymbol{e}_0 \\ \boldsymbol{\epsilon}_0 \\ \boldsymbol{e}_1 \\ \boldsymbol{\epsilon}_1 \\ \boldsymbol{e}_2 \end{bmatrix}}_{\underline{\boldsymbol{e}}_2}.$$

This system is $(M_0 + M_1 + M_2 + 2N) \times 3N$; we can form the least-squares estimate using

$$\begin{bmatrix} \widehat{\boldsymbol{x}}_{0|2} \\ \widehat{\boldsymbol{x}}_{1|2} \\ \widehat{\boldsymbol{x}}_{2|2} \end{bmatrix} = (\underline{\boldsymbol{A}}_2^{\mathrm{T}} \underline{\boldsymbol{A}}_2)^{-1} \underline{\boldsymbol{A}}_2^{\mathrm{T}} \underline{\boldsymbol{y}}_2.$$

In general, we have

$$\underline{\boldsymbol{A}}_k = \begin{bmatrix} \boldsymbol{A}_0 & & & & \\ \boldsymbol{F}_0 & -\mathbf{I} & & & \\ & \boldsymbol{A}_1 & & & \\ & \boldsymbol{F}_1 & & & \\ & & \ddots & & \\ & & & \boldsymbol{F}_{k-1} & -\mathbf{I} \\ & & & & \boldsymbol{A}_k \end{bmatrix}, \quad \underline{\boldsymbol{y}}_k = \begin{bmatrix} \boldsymbol{y}_0 \\ \mathbf{0} \\ \boldsymbol{y}_1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{y}_k \end{bmatrix}; \tag{1}$$

this system is $(M_0 + \cdots + M_k + kN) \times (k+1)N$, and we can for the least-squares estimate using

$$\begin{bmatrix} \widehat{\boldsymbol{x}}_{0|k} \\ \widehat{\boldsymbol{x}}_{1|k} \\ \vdots \\ \widehat{\boldsymbol{x}}_{k|k} \end{bmatrix} = (\underline{\boldsymbol{A}}_k^{\mathrm{T}} \underline{\boldsymbol{A}}_k)^{-1} \underline{\boldsymbol{A}}_k^{\mathrm{T}} \underline{\boldsymbol{y}}_k.$$

The $(k+1)N$-vector $\underline{\boldsymbol{A}}_k^{\mathrm{T}} \underline{\boldsymbol{y}}_k$ is

$$\underline{\boldsymbol{A}}_k^{\mathrm{T}} \underline{\boldsymbol{y}}_k = \begin{bmatrix} \boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{y}_0 \\ \boldsymbol{A}_1^{\mathrm{T}} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{A}_k^{\mathrm{T}} \boldsymbol{y}_k \end{bmatrix},$$

while the $(k+1)N \times (k+1)N$ matrix $\underline{\boldsymbol{A}}_k^{\mathrm{T}} \underline{\boldsymbol{A}}_k$ we have to invert is

$$\begin{bmatrix} \boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{A}_0 + \boldsymbol{F}_0^{\mathrm{T}} \boldsymbol{F}_0 & -\boldsymbol{F}_0^{\mathrm{T}} & \mathbf{0} & & & \\ -\boldsymbol{F}_0 & \mathbf{I} + \boldsymbol{A}_1^{\mathrm{T}} \boldsymbol{A}_1 + \boldsymbol{F}_1^{\mathrm{T}} \boldsymbol{F}_1 & -\boldsymbol{F}_1^{\mathrm{T}} & & & \\ \mathbf{0} & -\boldsymbol{F}_1 & \mathbf{I} + \boldsymbol{A}_2^{\mathrm{T}} \boldsymbol{A}_2 + \boldsymbol{F}_2^{\mathrm{T}} \boldsymbol{F}_2 & -\boldsymbol{F}_2^{\mathrm{T}} & & \\ \vdots & & & \ddots & & \\ & & & & \mathbf{I} + \boldsymbol{A}_{k-1}^{\mathrm{T}} \boldsymbol{A}_{k-1} + \boldsymbol{F}_{k-1}^{\mathrm{T}} \boldsymbol{F}_{k-1} & -\boldsymbol{F}_{k-1}^{\mathrm{T}} \\ & & & & -\boldsymbol{F}_{k-1} & \mathbf{I} + \boldsymbol{A}_k^{\mathrm{T}} \boldsymbol{A}_k \end{bmatrix} \tag{2}$$

Notice that this matrix is **block tridiagonal**; this structure is precisely what we will exploit in deriving the efficient update equations later on.

Example: Multiple measurements of a pulse

The following example illustrates the essential difference between the standard least-square framework and the Kalman filter.

We make three measurements of a patient's pulse, and want an (updated) estimate of its true value after each one. Our measurements are scalars, and have the form:

$$y = x + \text{noise}.$$

We record the three values y_0, y_1, y_2.

In the standard least-square framework, we solve the three problems

$$y_0 = 1\, x_* + e_0, \quad \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} x_* + \begin{bmatrix} e_0 \\ e_1 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} x_* + \begin{bmatrix} e_0 \\ e_1 \\ e_2 \end{bmatrix}.$$

If we call $\widehat{x}_k$ the least-squares estimate after recording y_k, a simple calculation shows that the estimate is simply the average of the measurements we have seen to date:

$$\widehat{x}_0 = y_0, \quad \widehat{x}_1 = \frac{y_0 + y_1}{2}, \quad \widehat{x}_2 = \frac{y_0 + y_1 + y_2}{3}.$$

Now suppose the we use the Kalman filtering framework to explicitly recognize that the pulse can "drift" over time. Our dynamical model is simple:

$$x_{k+1} = x_k + \epsilon_k.$$

This is saying that our best guess for the pulse at the next time step is that it takes the same value as before, but we are incorporating the fact that it will in fact not be exactly the same as this best guess.

In the Kalman framework, the three systems we solve look like

$$y_0 = 1\, x_0 + e_0,$$

$$\begin{bmatrix} y_0 \\ 0 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} + \begin{bmatrix} e_0 \\ \epsilon_0 \\ e_1 \end{bmatrix},$$

$$\begin{bmatrix} y_0 \\ 0 \\ y_1 \\ 0 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} e_0 \\ \epsilon_0 \\ e_1 \\ \epsilon_1 \\ e_2 \end{bmatrix}.$$

After the first measurement, our estimate for x_0 is the same:

$$\widehat{x}_{0|0} = y_0.$$

After the second measurement, we have

$$\begin{aligned} \begin{bmatrix} \widehat{x}_{0|1} \\ \widehat{x}_{1|1} \end{bmatrix} &= \left(\begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ 0 \\ y_1 \end{bmatrix} \\ &= \left(\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \right)^{-1} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \\ &= \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{2y_0+y_1}{3} \\ \frac{y_0+2y_1}{3} \end{bmatrix} \end{aligned}$$

So the best guess for where the pulse was is now a weighted average between the two samples; the estimate for x_1 weighs the most recent measurement more heavily than the first one. This is natural, since we are saying the pulse is changing between measurements.

After the third measurement, we have

$$\begin{bmatrix} \widehat{x}_{0|2} \\ \widehat{x}_{1|2} \\ \widehat{x}_{2|2} \end{bmatrix} = \left(\begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix} \right)^{-1} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$
$$= \begin{bmatrix} \frac{5y_0+2y_1+y_2}{8} \\ \frac{y_0+2y_1+y_2}{4} \\ \frac{y_0+2y_1+5y_2}{8} \end{bmatrix}.$$

Again, we see that as a direct result of allowing the pulse to drift, recent measurements are weighed more heavily in the current estimate.

Block tridiagonal factorization

To find the least-squares estimate of $x_0, \ldots, x_{N-1}$, we need to solve the $kN \times kN$ system given by (2) a few pages back. In this section, we will show how this this type of system allows a nice factorization which makes it clear how to solve it in a "streaming" manner.

Consider the following general symmetric block tridiagonal system:

$$\begin{bmatrix} \boldsymbol{D}_0 & \boldsymbol{C}_0^{\mathrm{T}} & \mathbf{0} & \cdots & & \mathbf{0} \\ \boldsymbol{C}_0 & \boldsymbol{D}_1 & \boldsymbol{C}_1^{\mathrm{T}} & \mathbf{0} & \cdots & \vdots \\ \mathbf{0} & \boldsymbol{C}_1 & \boldsymbol{D}_2 & \ddots & & \\ \vdots & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \boldsymbol{C}_{k-1}^{\mathrm{T}} \\ \mathbf{0} & \cdots & & & \boldsymbol{C}_{k-1} & \boldsymbol{D}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \vdots \\ \boldsymbol{x}_{k-1} \\ \boldsymbol{x}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \vdots \\ \vdots \\ \boldsymbol{b}_{k-1} \\ \boldsymbol{b}_k \end{bmatrix}$$

The system is $(k+1)N \times (k+1)N$, and each of the $\boldsymbol{D}_i$ and $\boldsymbol{C}_i$ are $N \times N$. Likewise, $\boldsymbol{x}_i \in \mathbb{R}^N$ and $\boldsymbol{b}_i \in \mathbb{R}^N$.

The matrix on the left-hand side above as the product of block lower triangular and block upper triangular matrices:

$$\begin{bmatrix} \boldsymbol{Q}_0 & \mathbf{0} & \cdots & & \mathbf{0} \\ \boldsymbol{C}_0 & \boldsymbol{Q}_1 & \mathbf{0} & & \\ \mathbf{0} & \boldsymbol{C}_1 & \boldsymbol{Q}_2 & \ddots & \\ \vdots & & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{C}_{k-1} & \boldsymbol{Q}_k \end{bmatrix} \begin{bmatrix} \mathbf{I} & \boldsymbol{U}_0 & \mathbf{0} & & \\ \mathbf{0} & \mathbf{I} & \boldsymbol{U}_1 & \mathbf{0} & \\ \vdots & & \ddots & \ddots & \\ & & & \ddots & \boldsymbol{U}_{k-1} \\ \mathbf{0} & & & \mathbf{0} & \mathbf{I} \end{bmatrix},$$

where the $\boldsymbol{C}_i$ are the same as in the original system, and the $\boldsymbol{Q}_i$ and $\boldsymbol{U}_i$ are also all $N \times N$. By inspecting the blocks, we see that we can compute the $\boldsymbol{Q}_i$ and $\boldsymbol{U}_i$ using the following iterative algorithm:

$\boldsymbol{Q}_0 = \boldsymbol{D}_0$
for $i = 1, 2, \ldots, k$
$\quad \boldsymbol{U}_{i-1} = \boldsymbol{Q}_{i-1}^{-1} \boldsymbol{C}_{i-1}^{\mathrm{T}}$
$\quad \boldsymbol{Q}_i = \boldsymbol{D}_i - \boldsymbol{C}_{i-1} \boldsymbol{U}_{i-1}$, meaning that $\boldsymbol{Q}_i = \boldsymbol{D}_i - \boldsymbol{C}_{i-1} \boldsymbol{Q}_{i-1}^{-1} \boldsymbol{C}_{i-1}^{\mathrm{T}}$
end

If the factorization is already in place, we can solve for the $\boldsymbol{x}_i$ by using forward substitution followed by back substitution. First, we solve

$$\begin{bmatrix} \boldsymbol{Q}_0 & \boldsymbol{0} & \cdots & & \boldsymbol{0} \\ \boldsymbol{C}_0 & \boldsymbol{Q}_1 & \boldsymbol{0} & & \\ \boldsymbol{0} & \boldsymbol{C}_1 & \boldsymbol{Q}_2 & \ddots & \\ \vdots & & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{C}_{k-1} & \boldsymbol{Q}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \\ \vdots \\ \\ \boldsymbol{w}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \vdots \\ \\ \boldsymbol{b}_k \end{bmatrix}$$

working from the top down:

$\boldsymbol{w}_0 = \boldsymbol{Q}_0^{-1} \boldsymbol{b}_0$
for $i = 1, 2, \ldots, k$
$\quad \boldsymbol{w}_i = \boldsymbol{Q}_i^{-1} (\boldsymbol{b}_i - \boldsymbol{C}_{i-1} \boldsymbol{w}_{i-1})$
end

With the $\boldsymbol{w}_i$ in hand, we can now solve

$$\begin{bmatrix} \mathbf{I} & \boldsymbol{U}_0 & \boldsymbol{0} & & \\ \boldsymbol{0} & \mathbf{I} & \boldsymbol{U}_1 & \boldsymbol{0} & \\ \vdots & & \ddots & \ddots & \\ & & & \ddots & \boldsymbol{U}_{k-1} \\ \boldsymbol{0} & & & \boldsymbol{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \\ \boldsymbol{x}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \\ \vdots \\ \\ \boldsymbol{w}_k \end{bmatrix}$$

for the $\boldsymbol{x}_i$ by working bottom up:

$\boldsymbol{x}_k = \boldsymbol{w}_k$
for $i = k-1, k-2, \ldots, 0$
$\quad \boldsymbol{x}_i = \boldsymbol{w}_i - \boldsymbol{U}_i \boldsymbol{x}_{i+1}$
end

Of course, we can combine the factorization with the forward step of the solve above to yield the following algorithm for solving symmetric[2] block tridiagonal systems:

Symmetric Block Tridiagonal Solve

Input: $N \times N$ matrices $\boldsymbol{D}_0, \ldots, \boldsymbol{D}_k$ and $\boldsymbol{C}_0, \ldots, \boldsymbol{C}_{k-1}$;
Right-hand side N-vectors $\boldsymbol{b}_0, \ldots, \boldsymbol{b}_k$

Initialize: $\boldsymbol{Q}_0 = \boldsymbol{D}_0$
$\boldsymbol{w}_0 = \boldsymbol{Q}_0^{-1} \boldsymbol{b}_0$

for $i = 1, 2, \ldots, k$ **do**

$\boldsymbol{U}_{i-1} = \boldsymbol{Q}_{i-1}^{-1} \boldsymbol{C}_{i-1}^{\mathrm{T}}$

$\boldsymbol{Q}_i = \boldsymbol{D}_i - \boldsymbol{C}_{i-1} \boldsymbol{U}_{i-1}$

$\boldsymbol{w}_i = \boldsymbol{Q}_i^{-1} (\boldsymbol{b}_i - \boldsymbol{C}_{i-1} \boldsymbol{w}_{i-1})$

end for

$\boldsymbol{x}_k = \boldsymbol{w}_k$

for $i = k-1, k-2, \ldots, 0$ **do**

$\boldsymbol{x}_i = \boldsymbol{w}_i - \boldsymbol{U}_i \boldsymbol{x}_{i+1}$

end for

[2] This is easily adapted to non-symmetric block tridiagonal, but we only need the symmetric case for what we do below.

Kalman filter update equations

Recalling equation (2) from earlier in these notes, we can find the least-squares solution $\widehat{\boldsymbol{x}}_{0|k}, \widehat{\boldsymbol{x}}_{1|k}, \ldots, \widehat{\boldsymbol{x}}_{k|k}$ at step k using the tridiagonal solver from the previous section with

$$\boldsymbol{C}_i = -\boldsymbol{F}_i, \quad i = 0, \ldots, k-1$$

$$\boldsymbol{D}_i = \begin{cases} \boldsymbol{A}_0^{\mathrm{T}} \boldsymbol{A}_0 + \boldsymbol{F}_0^{\mathrm{T}} \boldsymbol{F}_0, & i = 0 \\ \mathbf{I} + \boldsymbol{A}_i^{\mathrm{T}} \boldsymbol{A}_i + \boldsymbol{F}_i^{\mathrm{T}} \boldsymbol{F}_i, & i = 1, \ldots, k-1 \\ \mathbf{I} + \boldsymbol{A}_k^{\mathrm{T}} \boldsymbol{A}_k, & i = k. \end{cases}$$

That is great, but what is even better is that we can quickly move from the current best estimate $\widehat{\boldsymbol{x}}_{k|k}$ to the estimate at the next time step $\widehat{\boldsymbol{x}}_{k+1|k+1}$ very quickly.

Suppose that we have in our hands the solution to (2), consisting of the estimates

$$\widehat{\boldsymbol{x}}_{0|k}, \widehat{\boldsymbol{x}}_{1|k}, \ldots, \widehat{\boldsymbol{x}}_{k|k} \tag{3}$$

When the signal evolves ($\boldsymbol{x}_{k+1} = \boldsymbol{F}_k \boldsymbol{x}_k + \boldsymbol{\epsilon}_k$) and new measurements come in ($\boldsymbol{y}_{k+1} = \boldsymbol{A}_{k+1} \boldsymbol{x}_{k+1} + \boldsymbol{e}_{k+1}$), we break the update into two stages. The first is the "predict" stage, where we add the evolution equations — we solve,

$$\underline{\widetilde{\boldsymbol{A}}}_{k+1} = \begin{bmatrix} \boldsymbol{A}_0 & & & & & \\ \boldsymbol{F}_0 & -\mathbf{I} & & & & \\ & \boldsymbol{A}_1 & & & & \\ & \boldsymbol{F}_1 & & & & \\ & & \ddots & & & \\ & & & \boldsymbol{F}_{k-1} & -\mathbf{I} & \\ & & & & \boldsymbol{A}_k & \mathbf{0} \\ & & & & \boldsymbol{F}_k & -\mathbf{I} \end{bmatrix}, \quad \underline{\widetilde{\boldsymbol{y}}}_{k+1} = \begin{bmatrix} \boldsymbol{y}_0 \\ \mathbf{0} \\ \boldsymbol{y}_1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{y}_k \\ \mathbf{0} \end{bmatrix}.$$

This system has another block column, but since the last block column in non-zero only in the $k+1$ block, the $\boldsymbol{x}_{k+1}$ variables are essentially decoupled. The solution to $\underline{\widetilde{\boldsymbol{A}}}_{k+1}^{\mathrm{T}}\underline{\widetilde{\boldsymbol{A}}}_{k+1}\bar{\boldsymbol{x}} = \underline{\widetilde{\boldsymbol{A}}}_{k+1}^{\mathrm{T}}\underline{\widetilde{\boldsymbol{y}}}_{k+1}$ will be exactly the same as before, i.e. (3), but with an additional component given by

$$\widehat{\boldsymbol{x}}_{k+1|k} = \boldsymbol{F}_k\widehat{\boldsymbol{x}}_{k|k}.$$

We can interpret this as a **prediction** for $\widehat{\boldsymbol{x}}_{k+1|k+1}$ which we will update when we incorporate the measuremets $\boldsymbol{y}_{k+1}$.

The factorization is updated with

$$\begin{aligned}\widetilde{\boldsymbol{D}}_k &= \boldsymbol{D}_k + \boldsymbol{F}_k^{\mathrm{T}}\boldsymbol{F}_k = \mathbf{I} + \boldsymbol{A}_k^{\mathrm{T}}\boldsymbol{A}_k^{\mathrm{T}} + \boldsymbol{F}_k^{\mathrm{T}}\boldsymbol{F}_k \\ \boldsymbol{C}_k &= -\boldsymbol{F}_k \\ \widetilde{\boldsymbol{D}}_{k+1} &= \mathbf{I}.\end{aligned}$$

How exactly this works is thoroughly detailed in the Technical Details section below.

In the second stage, the "update" stage, we incorporate the measurements and solve

$$\underline{\boldsymbol{A}}_{k+1} = \begin{bmatrix} \boldsymbol{A}_0 & & & & & & \\ \boldsymbol{F}_0 & -\mathbf{I} & & & & & \\ & \boldsymbol{A}_1 & & & & & \\ & \boldsymbol{F}_1 & & & & & \\ & & \ddots & & & & \\ & & & \boldsymbol{F}_{k-1} & -\mathbf{I} & & \\ & & & & \boldsymbol{A}_k & \mathbf{0} & \\ & & & & \boldsymbol{F}_k & -\mathbf{I} & \\ & & & & \mathbf{0} & \boldsymbol{A}_{k+1} & \end{bmatrix}, \quad \underline{\boldsymbol{y}}_{k+1} = \begin{bmatrix} \boldsymbol{y}_0 \\ \mathbf{0} \\ \boldsymbol{y}_1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{y}_k \\ \mathbf{0} \\ \boldsymbol{y}_{k+1} \end{bmatrix}$$

This update works by changing the last block diagonal element to

$$\boldsymbol{D}_{k+1} = \widetilde{\boldsymbol{D}}_{k+1} + \boldsymbol{A}_{k+1}^{\mathrm{T}}\boldsymbol{A}_{k+1} = \mathbf{I} + \boldsymbol{A}_{k+1}^{\mathrm{T}}\boldsymbol{A}_{k+1}.$$

we can then quickly compute $\widehat{\boldsymbol{x}}_{k+1|k+1} = \boldsymbol{w}_{k+1}$ as

$$\widehat{\boldsymbol{x}}_{k+1|k+1} = \widehat{\boldsymbol{x}}_{k+1|k} + \boldsymbol{G}_{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{A}_{k+1}\widehat{\boldsymbol{x}}_{k+1|k}),$$

where $\boldsymbol{G}_{k+1}$ is the "Kalman gain" matrix (this is computed below). Notice that if the measurements $\boldsymbol{y}_{k+1}$ match our prediction $\widehat{\boldsymbol{x}}_{k+1|k}$, then we don't update this prediction at all — otherwise we pass the difference through the gain matrix $\boldsymbol{G}_{k+1}$ and add it to $\widehat{\boldsymbol{x}}_{k+1|k}$.

Here are the updating equations, which are carefully derived in the Technical Details section:

Kalman Filter

Initialize: $\widehat{\boldsymbol{x}}_{0|0} = (\boldsymbol{A}_0^{\mathrm{T}}\boldsymbol{A}_0)^{-1}\boldsymbol{A}_0^{\mathrm{T}}\boldsymbol{y}_0$

$\boldsymbol{P}_{0|0} = (\boldsymbol{A}_0^{\mathrm{T}}\boldsymbol{A}_0)^{-1}$

for $k = 0, 1, 2, \ldots$ **do**

State update extrapolation: $\widehat{\boldsymbol{x}}_{k+1|k} = \boldsymbol{F}_k\widehat{\boldsymbol{x}}_{k|k}$

Info matrix extrapolation: $\boldsymbol{P}_{k+1|k} = \boldsymbol{F}_k\boldsymbol{P}_{k|k}\boldsymbol{F}_k^{\mathrm{T}} + \mathbf{I}$

Kalman gain: $\boldsymbol{G}_{k+1} = \boldsymbol{P}_{k+1|k}\boldsymbol{A}_{k+1}^{\mathrm{T}}(\boldsymbol{A}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{A}_{k+1}^{\mathrm{T}} + \mathbf{I})^{-1}$

State update: $\widehat{\boldsymbol{x}}_{k+1|k+1} = \widehat{\boldsymbol{x}}_{k+1|k} + \boldsymbol{G}_{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{A}_{k+1}\widehat{\boldsymbol{x}}_{k+1|k})$

Info matrix update: $\boldsymbol{P}_{k+1|k+1} = (\mathbf{I} - \boldsymbol{G}_{k+1}\boldsymbol{A}_{k+1})\boldsymbol{P}_{k+1|k}$

end for

So the cost of moving from $\widehat{\boldsymbol{x}}_{k|k}$ to $\widehat{\boldsymbol{x}}_{k+1|k+1}$ is $O(N^3) + O(M_{k+1}N^2)$, which is about the same as it would cost to simply compute the standard least-squares solution for $\boldsymbol{x}_{k+1}$ using only $\boldsymbol{y}_{k+1}$.

Technical Details: Matrix Inversion Lemma

The general statement of the *Sherman-Morrison-Woodbury* identity is that

$$(\boldsymbol{W} + \boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{Z})^{-1} = \boldsymbol{W}^{-1} - \boldsymbol{W}^{-1}\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{Y}^{-1} + \boldsymbol{Z}\boldsymbol{W}^{-1}\boldsymbol{X}^{\mathrm{T}})^{-1}\boldsymbol{Z}\boldsymbol{W}^{-1}$$

where $\boldsymbol{W}$ is $N \times N$ and invertible, $\boldsymbol{X}$ and $\boldsymbol{Z}$ are $R \times N$, and $\boldsymbol{Y}$ is $R \times R$ and invertible.

The proof of this is straightforward. Given any right hand side $\boldsymbol{v} \in \mathbb{R}^N$, we would like to solve

$$(\boldsymbol{W} + \boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{Z})\boldsymbol{w} = \boldsymbol{v} \tag{4}$$

for $\boldsymbol{w}$. Set

$$\boldsymbol{z} = \boldsymbol{Y}\boldsymbol{Z}\boldsymbol{w} \quad \Rightarrow \quad \boldsymbol{Y}^{-1}\boldsymbol{z} = \boldsymbol{Z}\boldsymbol{w}.$$

We now have the set of two equations

$$\begin{aligned} \boldsymbol{W}\boldsymbol{w} + \boldsymbol{X}^{\mathrm{T}}\boldsymbol{z} &= \boldsymbol{v} \\ \boldsymbol{Z}\boldsymbol{w} - \boldsymbol{Y}^{-1}\boldsymbol{z} &= \boldsymbol{0}. \end{aligned}$$

Manipulating the first equation yields

$$\boldsymbol{w} = \boldsymbol{W}^{-1}(\boldsymbol{v} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{z}), \tag{5}$$

and then plugging this into the second equation gives us

$$\begin{aligned} \boldsymbol{Z}\boldsymbol{W}^{-1}\boldsymbol{v} - \boldsymbol{Z}\boldsymbol{W}^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{z} - \boldsymbol{Y}^{-1}\boldsymbol{z} &= \boldsymbol{0} \\ \Rightarrow \quad \boldsymbol{z} &= (\boldsymbol{Y}^{-1} + \boldsymbol{Z}\boldsymbol{W}^{-1}\boldsymbol{X}^{\mathrm{T}})^{-1}\boldsymbol{Z}\boldsymbol{W}^{-1}\boldsymbol{v}. \end{aligned} \tag{6}$$

So then given any $\boldsymbol{v} \in \mathbb{R}^N$, we can solve for $\boldsymbol{w}$ in (4) by combining (5) and (6) to get

$$\boldsymbol{w} = \boldsymbol{W}^{-1}\boldsymbol{v} - \boldsymbol{W}^{-1}\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{Y}^{-1} + \boldsymbol{Z}\boldsymbol{W}^{-1}\boldsymbol{X}^{\mathrm{T}})^{-1}\boldsymbol{Z}\boldsymbol{W}^{-1}\boldsymbol{v}.$$

As this holds for any right-hand side $\boldsymbol{v}$, this establishes the result.

Technical Details: Kalman Filter Updates

For the first stage of the update, the "predict stage", the system of equations moves from $\underline{\boldsymbol{A}}_k$ in (1) to

$$\widetilde{\underline{\boldsymbol{A}}}_{k+1} = \begin{bmatrix} \boldsymbol{A}_0 & & & & & & \\ \boldsymbol{F}_0 & -\mathbf{I} & & & & & \\ & \boldsymbol{A}_1 & & & & & \\ & \boldsymbol{F}_1 & & & & & \\ & & \ddots & & & & \\ & & & \boldsymbol{F}_{k-1} & -\mathbf{I} & & \\ & & & & \boldsymbol{A}_k & \mathbf{0} & \\ & & & & \boldsymbol{F}_k & -\mathbf{I} & \end{bmatrix}, \quad \widetilde{\underline{\boldsymbol{y}}}_{k+1} = \begin{bmatrix} \boldsymbol{y}_0 \\ \mathbf{0} \\ \boldsymbol{y}_1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{y}_k \\ \mathbf{0} \end{bmatrix}$$

The least-squares solution to this system will be unchanged, except for an additional term in the $k+1$st block, which we call $\widehat{\boldsymbol{x}}_{k+1|k}$.

With our previous factorization in hand, we have

$$\boldsymbol{P}_{k|k} := \boldsymbol{Q}_k^{-1}$$

computed already. The addition of the $\boldsymbol{F}_k$ above changes the $\boldsymbol{D}_k$ in our block tridiagonal system to

$$\widetilde{\boldsymbol{D}}_k = \boldsymbol{D}_k + \boldsymbol{F}_k^{\mathrm{T}} \boldsymbol{F}_k,$$

and so the factorization is updated with

$$\begin{aligned} \widetilde{\boldsymbol{Q}}_k &= \boldsymbol{Q}_k + \boldsymbol{F}_k^{\mathrm{T}} \boldsymbol{F}_k \\ \widetilde{\boldsymbol{w}}_k &= \widetilde{\boldsymbol{Q}}_k^{-1} (\boldsymbol{b}_k + \boldsymbol{F}_{k-1} \boldsymbol{w}_{k-1}). \end{aligned}$$

This new system also has $k+1$ blocks, so we need to compute the new terms in our running factorization: $\boldsymbol{U}_k, \widetilde{\boldsymbol{Q}}_{k+1}$, and $\widetilde{\boldsymbol{w}}_{k+1}$.

By the matrix inversion lemma

$$\widetilde{\boldsymbol{Q}}_k^{-1} = \boldsymbol{Q}_k^{-1} - \boldsymbol{Q}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}}(\mathbf{I} + \boldsymbol{F}_k\boldsymbol{Q}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}})^{-1}\boldsymbol{F}_k\boldsymbol{Q}_k^{-1},$$

so we can rewrite $\widetilde{\boldsymbol{w}}_k$ as

$$\widetilde{\boldsymbol{w}}_k = \boldsymbol{w}_k - \boldsymbol{Q}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}}(\mathbf{I} + \boldsymbol{F}_k\boldsymbol{Q}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}})^{-1}\boldsymbol{F}_k\boldsymbol{w}_k.$$

We now have $\boldsymbol{C}_k = -\boldsymbol{F}_k$ and $\widetilde{\boldsymbol{D}}_{k+1} = \mathbf{I}$, and $\widetilde{\boldsymbol{b}}_{k+1} = \mathbf{0}$, so we move forward with the running factorization by solving

$$\begin{aligned}
\boldsymbol{U}_k &= -\widetilde{\boldsymbol{Q}}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}} \\
\widetilde{\boldsymbol{Q}}_{k+1} &= \mathbf{I} - \boldsymbol{F}_k\widetilde{\boldsymbol{Q}}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}} \\
\widetilde{\boldsymbol{w}}_{k+1} &= \widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{F}_k\widetilde{\boldsymbol{w}}_k \\
&= \widetilde{\boldsymbol{Q}}_{k+1}^{-1}(\mathbf{I} - \boldsymbol{F}_k\boldsymbol{Q}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}}(\mathbf{I} + \boldsymbol{F}_k\boldsymbol{Q}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}})^{-1})\boldsymbol{F}_k\boldsymbol{w}_k \\
&= \widetilde{\boldsymbol{Q}}_{k+1}^{-1}(\mathbf{I} + \boldsymbol{F}_k\boldsymbol{Q}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}})^{-1}\boldsymbol{F}_k\boldsymbol{w}_k \qquad (7) \\
&= \boldsymbol{F}_k\boldsymbol{w}_k \qquad (8)
\end{aligned}$$

where the second to last step (7) follows from the identity for symmetric matrices

$$\mathbf{I} - \boldsymbol{A}(\mathbf{I} + \boldsymbol{A})^{-1} = (\mathbf{I} + \boldsymbol{A})^{-1}, \qquad (9)$$

(just multiply both sides on the right by $(\mathbf{I} + \boldsymbol{A})$), and the last step (8) follows from an application of the matrix inversion lemma

$$\begin{aligned}
(\mathbf{I} + \boldsymbol{F}_k\boldsymbol{Q}_k^{-1}\boldsymbol{F}_k^{\mathrm{T}})^{-1} &= \mathbf{I} - \boldsymbol{F}_k(\boldsymbol{Q}_k + \boldsymbol{F}_k^{\mathrm{T}}\boldsymbol{F}_k)^{-1}\boldsymbol{F}_k^{\mathrm{T}} \\
&= \mathbf{I} - \boldsymbol{F}_k\widetilde{\boldsymbol{Q}}_k^{-1}\boldsymbol{F}_k \\
&= \widetilde{\boldsymbol{Q}}_{k+1}.
\end{aligned}$$

Since $\widetilde{\boldsymbol{w}}_{k+1} = \widehat{\boldsymbol{x}}_{k+1|k}$ and $\boldsymbol{w}_k = \widehat{\boldsymbol{x}}_{k|k}$, we can rewrite the conclusion above as

$$\widehat{\boldsymbol{x}}_{k+1|k} = \boldsymbol{F}_k \widehat{\boldsymbol{x}}_{k|k}.$$

We will also denote

$$\begin{aligned} \boldsymbol{P}_{k+1|k} := \widetilde{\boldsymbol{Q}}_{k+1}^{-1} &= \mathbf{I} + \boldsymbol{F}_k \boldsymbol{Q}_k^{-1} \boldsymbol{F}_k \\ &= \mathbf{I} + \boldsymbol{F}_k \boldsymbol{P}_{k|k} \boldsymbol{F}_k. \end{aligned}$$

We now go to the second stage of the update, where we add the measurements $\boldsymbol{y}_{k+1} = \boldsymbol{A}_{k+1}\boldsymbol{x}_{x+1} + \boldsymbol{e}_{k+1}$. We add a block row to our system of equations; we now want to solve

$$\underline{\boldsymbol{A}}_{k+1} = \begin{bmatrix} \boldsymbol{A}_0 & & & & & \\ \boldsymbol{F}_0 & -\mathbf{I} & & & & \\ & \boldsymbol{A}_1 & & & & \\ & \boldsymbol{F}_1 & & & & \\ & & \ddots & & & \\ & & & \boldsymbol{F}_{k-1} & -\mathbf{I} & \\ & & & & \boldsymbol{A}_k & \mathbf{0} \\ & & & & \boldsymbol{F}_k & -\mathbf{I} \\ & & & & \mathbf{0} & \boldsymbol{A}_{k+1} \end{bmatrix}, \quad \underline{\boldsymbol{y}}_{k+1} = \begin{bmatrix} \boldsymbol{y}_0 \\ \mathbf{0} \\ \boldsymbol{y}_1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{y}_k \\ \mathbf{0} \\ \boldsymbol{y}_{k+1} \end{bmatrix}$$

This means that we are replacing $\widetilde{\boldsymbol{D}}_{k+1} = \mathbf{I}$ with $\boldsymbol{D}_{k+1} = \mathbf{I} + \boldsymbol{A}_{k+1}^{\mathrm{T}} \boldsymbol{A}_{k+1}$ and $\widetilde{\boldsymbol{b}}_{k+1} = \mathbf{0}$ with $\boldsymbol{b}_{k+1} = \boldsymbol{A}_{k+1}^{\mathrm{T}} \boldsymbol{y}_{k+1}$. We update the factor-and-solve as follows:

$$\begin{aligned} \boldsymbol{D}_{k+1} &= \mathbf{I} + \boldsymbol{A}_{k+1}^{\mathrm{T}} \boldsymbol{A}_{k+1} \\ \boldsymbol{Q}_{k+1} &= \widetilde{\boldsymbol{Q}}_{k+1} + \boldsymbol{A}_{k+1}^{\mathrm{T}} \boldsymbol{A}_{k+1} \\ \boldsymbol{w}_{k+1} &= \boldsymbol{Q}_{k+1}^{-1} (\boldsymbol{A}_{k+1}^{\mathrm{T}} \boldsymbol{y}_{k+1} + \boldsymbol{F}_k \widetilde{\boldsymbol{w}}_k). \end{aligned} \tag{10}$$

Using the matrix-inversion lemma, we can write

$$\boldsymbol{Q}_{k+1}^{-1} = \widetilde{\boldsymbol{Q}}_{k+1}^{-1} - \widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}}(\mathbf{I} + \boldsymbol{A}_{k+1}\widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}})^{-1}\boldsymbol{A}_{k+1}\widetilde{\boldsymbol{Q}}_{k+1}^{-1}, \quad (11)$$

and so using the fact from above that $\widetilde{\boldsymbol{w}}_{k+1} = \widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{F}_k\widetilde{\boldsymbol{w}}_k = \widetilde{\boldsymbol{Q}}_{k+1}^{-1}\widehat{\boldsymbol{x}}_{k+1|k}$, the two terms in the expression for $\boldsymbol{w}_{k+1} = \widehat{\boldsymbol{x}}_{k+1|k+1}$ becomes

$$\boldsymbol{Q}_{k+1}^{-1}\boldsymbol{F}_k\widetilde{\boldsymbol{w}}_k = \widehat{\boldsymbol{x}}_{k+1|k} - \widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}}(\mathbf{I} + \boldsymbol{A}_{k+1}\widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}})^{-1}\boldsymbol{A}_{k+1}\widehat{\boldsymbol{x}}_{k+1|k} \quad (12)$$

and

$$\begin{aligned}\boldsymbol{Q}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}}\boldsymbol{y}_{k+1} = & \\ \widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}} & \left(\mathbf{I} - (\mathbf{I} + \boldsymbol{A}_{k+1}\widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}})^{-1}\boldsymbol{A}_{k+1}\widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}}\right)\boldsymbol{y}_{k+1} \\ & = \widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}}\left(\mathbf{I} + \boldsymbol{A}_{k+1}\widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}}\right)^{-1}\boldsymbol{y}_{k+1}, \quad (13)\end{aligned}$$

where we have again used the identity (9). Combining (12) and (13) with (10), we have

$$\boldsymbol{w}_{k+1} = \widehat{\boldsymbol{x}}_{k+1|k+1} = \widehat{\boldsymbol{x}}_{k+1|k} + \boldsymbol{G}_{k+1}\left(\boldsymbol{y}_{k+1} - \boldsymbol{A}_{k+1}\widehat{\boldsymbol{x}}_{k+1|k}\right),$$

where

$$\begin{aligned}\boldsymbol{G}_{k+1} &= \widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}}\left(\mathbf{I} + \boldsymbol{A}_{k+1}\widetilde{\boldsymbol{Q}}_{k+1}^{-1}\boldsymbol{A}_{k+1}^{\mathrm{T}}\right)^{-1} \\ &= \boldsymbol{P}_{k+1|k}\boldsymbol{A}_{k+1}^{\mathrm{T}}\left(\mathbf{I} + \boldsymbol{A}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{A}_{k+1}^{\mathrm{T}}\right)^{-1}.\end{aligned}$$

Finally, (11) above also gives us the update

$$\boldsymbol{P}_{k+1|k+1} := \boldsymbol{Q}_{k+1}^{-1} = \left(\mathbf{I} - \boldsymbol{G}_{k+1}\boldsymbol{A}_{k+1}\right)\boldsymbol{P}_{k+1|k}.$$