How do we update estimates based on current estimates?

Let's start with 2 "batches":

$$\tilde{y}_1 = [\tilde{y}_{11} \quad \tilde{y}_{12} \quad \cdots \quad \tilde{y}_{1m_1}]^T$$

$$\tilde{y}_2 = [\tilde{y}_{21} \quad \tilde{y}_{22} \quad \cdots \quad \tilde{y}_{2m_2}]^T$$

$$\tilde{y}_1 = H_1 \underline{x} + \underline{v}_1$$

$$\tilde{y}_2 = H_2 \underline{x} + v_2$$

trying to estimate the $\underline{x}$

For $\tilde{y}_1$, from LLS before we have:

$$\hat{\underline{x}}_1 = (H_1^T W_1 H_1)^{-1} H_1^T W_1 \tilde{y}_1$$

If we had both measurements together:

$$\tilde{\underline{y}} = H \underline{x} + \underline{v}$$

$$\begin{bmatrix} \tilde{\underline{y}}_1 \\ \tilde{\underline{y}}_2 \end{bmatrix} = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} x + \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \end{bmatrix}$$

Let's assume:

$$W = \left[\begin{array}{c|c} W_1 & 0 \\ \hline 0 & W_2 \end{array}\right]$$

ie. there is no correlation of noise btwn first & second set of measurements

Given both measurement sets, we have:

$$\hat{\underline{x}}_2 = (H^T W H)^{-1} H^T W \tilde{\underline{y}}$$

$\hat{\underline{x}}_2$ takes into account BOTH measurement sets

Since $W$ is block diagonal, we can write:

$$\hat{\underline{x}}_2 = \left[H_1^T W_1 H_1 + H_2^T W_2 H_2\right]^{-1} \left(H_1^T W_1 \tilde{\underline{y}}_1 + H_2^T W_2 \tilde{\underline{y}}_2\right)$$

But we don't want to recalculate everything every time - we want efficient use of previous calculations.

define:

$$P_1 = \left[H_1^T W_1 H_1\right]^{-1}$$

$$P_2 = \left[H_1^T W_1 H_1 + H_2^T W_2 H_2\right]^{-1}$$

thus:

$$P_2^{-1} = P_1^{-1} + H_2^T W_2 H_2$$

Now we can rewrite:

$$\hat{\underline{x}}_1 = P_1 H_1^T W_1 \tilde{y}_1$$

$$\hat{\underline{x}}_2 = P_2 \left(\underbrace{H_1^T W_1 \tilde{y}_1} + H_2^T W_2 \tilde{y}_2\right)$$

We want to replace this with $\hat{\underline{x}}_1$, because we already did those calculations

$$H_1^T W_1 \tilde{y}_1 = P_1^{-1} \hat{\underline{x}}_1$$

Substituting that in:

$$\hat{\underline{x}}_2 = P_2\left(P_1^{-1}\hat{\underline{x}}_1 + H_2^T W_2 \tilde{y}_2\right)$$

$$\hat{\underline{x}}_2 = P_2 P_1^{-1} \hat{\underline{x}}_1 + P_2 H_2^T W_2 \tilde{y}_2$$

inverses are expensive, so use:

$$P_2^{-1} = P_1^{-1} + H_2^T W_2 H_2$$ from earlier

$$\hat{\underline{x}}_2 = P_2\left[P_2^{-1} - H_2^T W_2 H_2\right]\hat{\underline{x}}_1 + P_2 H_2^T W_2 \tilde{y}_2$$

$$= \left[I - P_2 H_2^T W_2 H_2\right]\hat{\underline{x}}_1 + P_2 H_2^T W_2 \tilde{y}_2$$

$$= \hat{\underline{x}}_1 - \underline{P_2 H_2^T W_2} H_2 \hat{\underline{x}}_1 + \underline{P_2 H_2^T W_2} \tilde{y}_2$$

$$= \underline{\hat{x}}_1 + \underbrace{P_2 H_2^T W_2}_{\text{"K"}} \left[ \underbrace{\tilde{y}_2 - H_2 \underline{\hat{x}}_1} \right]$$

previous "a priori" estimate of $\underline{\hat{x}}$

"gain" matrix

"error" between new measurements and expectation based on previous estimate

Now we can generalize:

**Kalman update equations**

$$\underline{\hat{x}}_{k+1} = \underline{\hat{x}}_k + K_{k+1} \left( \tilde{y}_{k+1} - H_{k+1} \underline{\hat{x}}_k \right)$$

$$K_{k+1} = P_{k+1} H_{k+1}^T W_{k+1}$$

$$P_{k+1}^{-1} = P_k^{-1} + H_{k+1}^T W_{k+1} H_{k+1}$$

This is a time-varying dynamic system - rearranging yields:

$$\underline{\hat{x}}_{k+1} = \underbrace{\left[ I - K_{k+1} H_{k+1} \right]}_{\text{you can check stability, response time, etc.}} \underline{\hat{x}}_{k} + \left[ K_{k+1} \right] \underline{\tilde{y}}_{k+1}$$

This calculation requires an inverse (expensive)

$$P_{k+1} = \left[ P_k^{-1} + H_{k+1}^T W_{k+1} H_{k+1} \right]^{-1}$$

But, if the new measurements being added are few compared to the number of states (often the case) we can rewrite the update eqn:

$$P_{k+1} = P_k - P_k H_{k+1}^T \left( H_{k+1} P_k H_{k+1}^T + W_{k+1}^{-1} \right)^{-1} H_{k+1} P_k$$

Using Sherman-Morrison-Woodbury matrix inversion lemma. (1.69)

if $H$ is, for example $1 \times n$, then $W$ is $1 \times 1$, so we end up inverting a $1 \times 1$ matrix instead of an $n \times n$.

# Covariance form

You can also move the inverse to the kalman gain eqn:

$$K_{k+1} = P_k H_{k+1}^T \left[ H_{k+1} P_k H_{k+1}^T + W_{k+1}^{-1} \right]^{-1} \quad (1.79)$$

which is also a smaller inversion

$$P_{k+1} = \left[ I - K_{k+1} H_{k+1} \right] P_k \quad (1.80)$$

This is the most common form, and what we see in the kalman filter.

However, it involves an inverse, and $\left[ I - K_{k+1} H_{k+1} \right]$ can result in non symmetric matrices b/c of numerics.

# Inverses & Stability

## Sequential updates

Dealing with 1 measurement at a time turns the inverses into divisions

## Joseph form

We can resolve those issues with:

$$S_{k+1} K_{k+1} = P_k H_{k+1}^T \quad ,$$ where:

$$S_{k+1} = \left[ H_{k+1} P_k H_{k+1}^T + W_{k+1} \right]^{-1},$$

and solve for $K_{k+1}$ as a linear system: use:

- Matlab's "\" command
- Python: scipy.linalg.solve

These commands solve the linear sys. without taking an inverse.

How? Depends... e.g. LU or QR decomposition.

$$P_{k+1} = \left[ I - K_{k+1} H_{k+1} \right] P_k \left[ I - K_{k+1} H_{k+1} \right]^T + K_{k+1} W_{k+1} K_{k+1}^T$$

which guarantees that $P$ remains positive definite.

# Square-root filters

Keep track of $S$, where $P = SS^T$ instead of $P$.

Use QR decomposition or Cholesky updates instead of inversions

Think of it as a built in "\" operation.

# Information form

You can also work with information ($P^{-1}$) and the information vector ($P^{-1}x$). Less common.