Lecture 8: Recursive Square-Root & QR Updating for Least Squares and Kalman Filtering

Prof. Krishna R. Pattipati
Dept. of Electrical and Computer Engineering
University of Connecticut
Contact: krishna@engr.uconn.edu (860) 486-2890

ECE 6435

Adv Numerical Methods in Sci Comp

Fall 2008

October 15, 2008

Lecture Outline

- Recursive (sequential) Least Squares
- Sequential LDL^T Factorization updates
- Sequential QR updates
- Application to Kalman filtering

Recursive Least Squares (RLS)

- Suppose we have measurements $b_i = \underline{a}_i^T \underline{x},\ b_i$ scalar for $i = 1, 2, \ldots, k,\ \underline{x} \in R^n$ unknowns (occurs in many applications, e.g., fitting an n^{th} order Polynomial

$$\begin{bmatrix} \leftarrow \underline{a}_1^T \rightarrow \\ \leftarrow \underline{a}_2^T \rightarrow \\ . \\ . \\ \leftarrow \underline{a}_k^T \rightarrow \end{bmatrix} \underline{x} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_k \end{bmatrix}$$ assume $k > n$ and A has full column rank n

- Objective is to find $\underline{x}_{\mathrm{LS}}$ to minimize the mean-squared error (MMSE)

MSE: $\left\| A_k \underline{x} - \underline{b}_k \right\|_2^2$

MMSE: Find $\underline{x}_{\mathrm{LS}} \ni \left\| A_k \underline{x} - \underline{b}_k \right\|_2^2$ is a minimum

Know $\underline{\hat{x}}_{\mathrm{LS}} = A_k^{\dagger} \underline{b}_k = \left(A_k^T A_k \right)^{-1} A_k^T \underline{b}_k \triangleq \underline{\hat{x}}_k$

Setting the Stage

- Suppose now we make a (k+1)st measurement $b_{k+1} = \underline{a}_{k+1}^T \underline{x}$

- **Q:** Can we update our previous estimates in light of b_{k+1} without recomputing $A_{k+1}^T A_{k+1}$ or Householder or Givens or Gram-Schmidt?

- **A:** Yes! This is precisely what is done in RLS, Kalman filtering, etc.

- How does recursive least-squares (RLS) work ?

  - Let $\hat{\underline{x}}_k$ be an estimate of $\underline{x}$ using $b_1, b_2, \ldots, b_k = \left(A_k^T A_k\right)^{-1} A_k^T \underline{b}_k$
  - Let $\hat{\underline{x}}_{k+1}$ be an estimate of $\underline{x}$ using $b_1, b_2, \ldots, b_{k+1}$

    $$\hat{\underline{x}}_{k+1} = \left(A_{k+1}^T A_{k+1}\right)^{-1} A_{k+1}^T \underline{b}_{k+1}$$

  - Define $P_{k+1} = \left(A_{k+1}^T A_{k+1}\right)^{-1}$ and $P_k = \left(A_k^T A_k\right)^{-1}$
  - In RLS, we estimate $\hat{\underline{x}}_{k+1}$ from $\hat{\underline{x}}_k$ and b_{k+1}, and P_{k+1} from P_k and $\underline{a}_{k+1}$

Sequential Update of Covariance Matrix

- Mechanics of RLS process
  - Consider $A_{k+1}A_{k+1}^T$:

$$A_{k+1}^T A_{k+1} = A_k^T A_k + \underline{a}_{k+1}\underline{a}_{k+1}^T$$

$$\Rightarrow P_{k+1}^{-1} = P_k^{-1} + \underline{a}_{k+1}\underline{a}_{k+1}^T \quad ; P_k^{-1} \sim \text{is the so called information matrix}$$

so, every succesive measurement adds "INFORMATION"

  - Key: Sherman-Morrison-Woodbury Formula

- Consider three matrices: A is *n* x *n*, B is *n* x *m* C = *n* x *m*
  - Then Sherman-Morrison-Woodbury Formula gives:

$$\left(A + BC^T\right)^{-1} = A^{-1} - A^{-1}B\left(I + C^T A^{-1} B\right)^{-1} C^T A^{-1}$$

$$\Rightarrow P_{k+1} = P_k - P_k \underline{a}_{k+1}\left(\underline{a}_{k+1}^T P_k \underline{a}_{k+1} + 1\right)^{-1} \underline{a}_{k+1}^T P_k$$

$\Rightarrow$ Requires scalar inversion

$$\Rightarrow P_{k+1} = P_k - P_k \underline{a}_{k+1}\underline{a}_{k+1}^T P_k / \left(1 + \underline{a}_{k+1}^T P_k \underline{a}_{k+1}\right)$$

Sequential Update of Estimate

- To compute $\hat{\underline{x}}_{k+1}$

$$\hat{\underline{x}}_{k+1} = P_{k+1}\begin{bmatrix} A_k^T & \underline{a}_{k+1} \end{bmatrix}\begin{bmatrix} \underline{b}_k \\ b_{k+1} \end{bmatrix} = P_{k+1}\left[A_k^T \underline{b}_k + b_{k+1}\underline{a}_{k+1} \right]$$

$$\hat{\underline{x}}_{k+1} = \left[P_k - \frac{P_k \underline{a}_{k+1}\underline{a}_{k+1}^T P_k}{1+\underline{a}_{k+1}^T P_k \underline{a}_{k+1}} \right]\left[A_k^T \underline{b}_k + b_{k+1}\underline{a}_{k+1} \right]$$

$$= \hat{\underline{x}}_k + P_k \underline{a}_{k+1} b_{k+1} - \frac{P_k \underline{a}_{k+1}\underline{a}_{k+1}^T P_k \underline{a}_{k+1}}{1+\underline{a}_{k+1}^T P_k \underline{a}_{k+1}} b_{k+1} - \frac{P_k \underline{a}_{k+1}\underline{a}_{k+1}^T \overbrace{P_k A_k^T \underline{b}_k}^{\hat{\underline{x}}_k}}{1+\underline{a}_{k+1}^T P_k \underline{a}_{k+1}}$$

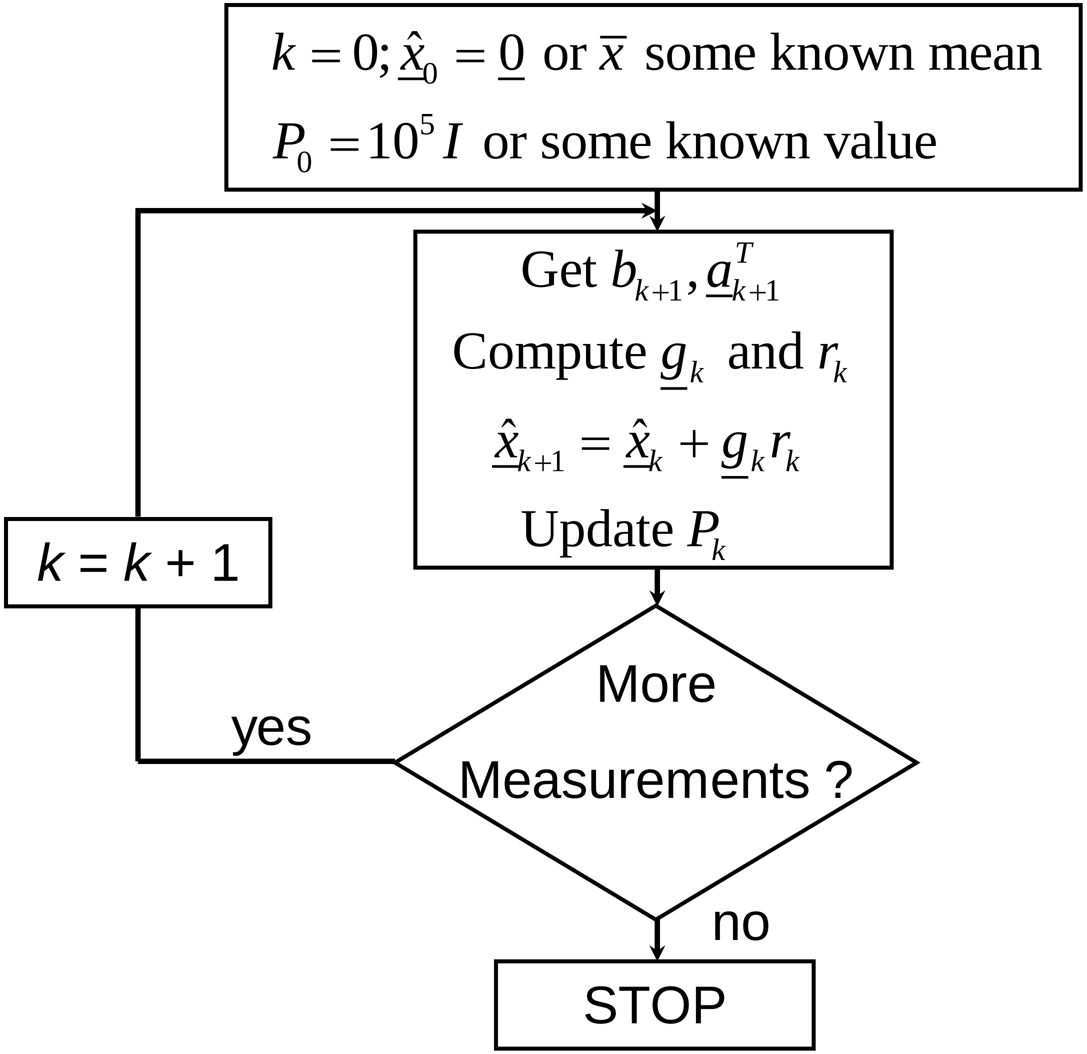
$$= \hat{\underline{x}}_k + \underbrace{\frac{P_k \underline{a}_{k+1}}{1+\underline{a}_{k+1}^T P_k \underline{a}_{k+1}}}_{\text{Gain vector},\ \underline{g}_k}\underbrace{\left[b_{k+1} - \underline{a}_{k+1}^T \hat{\underline{x}}_k \right]}_{\text{Residual or innovation},\ r_k}$$

$$= \hat{\underline{x}}_k + \underline{g}_k r_k = \left[I - \underline{g}_k \underline{a}_{k+1}^T \right]\hat{\underline{x}}_k + \underline{g}_k b_{k+1}$$

$\Rightarrow$ $\hat{\underline{x}}_{k+1}$ is a weighted sum of previous estimate and current measurement

$\Rightarrow$ This is similar to the measurement update of a Kalman filter

RLS is Simple to Implement



Round-off Error Issues

- Major problem: Negative sign in P_k equation causes P_k to go indefinite due to round-off errors (e.g., negative diagonals)

- Other formulae to overcome indefiniteness
  1. Joseph's form:

$$P_{k+1} = \left(I - \underline{g}_k \underline{a}_{k+1}^T\right) P_{k+1} \left(I - \underline{g}_k \underline{a}_{k+1}^T\right)^T + \underline{g}_k \underline{g}_k^T$$

     This transformation requires twice the number of operations over the ordinary RLS

  2. Square-root or LDL^T update
     – Idea: force P_k and P_{k+1} to be PD

$$\text{i.e., write } \quad P_k = L_k D_k L_k^T \text{ via "LDL}^T\text{" factorization}$$

$$L_k = \text{ unit lower } \Delta;$$

$$D_k = diag\left(d_i\right), d_i > 0$$

Setting the Stage for LDL^T Update

- *Q*: Can we go from $\begin{bmatrix} L_k \\ D_k \end{bmatrix} \rightarrow \begin{bmatrix} L_{k+1} \\ D_{k+1} \end{bmatrix}$ recursively ?
- *A*: Yes, but slightly complicated
- Simplicity of notation, let

$$P_k = LDL^T, \quad P_{k+1} = \bar{L}\bar{D}\bar{L}^T, \quad \underline{a}_{k+1} = \underline{a}, \text{ then}$$

$$P_{k+1} = LDL^T - \left(LDL^T \underline{a}\,\underline{a}^T LDL^T\right) / \left(1 + \underline{a}^T LDL^T \underline{a}\right)$$

- To simplify the expression for P_{k+1}, let $\underline{f} = L^T \underline{a}$, then

$$P_{k+1} = L\left[D - \frac{\underline{v}\,\underline{v}^T}{1 + \underline{f}^T D \underline{f}}\right] L^T; \underline{v} = D\underline{f} \text{ or } v_i = d_i f_i$$

- So, if we can find $\tilde{L}\tilde{D}\tilde{L}^T$ of the terms in brackets, then we have solved the problem:

$$\bar{L} = L\tilde{L} \text{ and } \bar{D} = \tilde{D}$$

LDL^T Update with Rank 1 Correction

- This is a special case of the following more general problem:

  "Given $A = LDL^T$, find $\bar{L}\bar{D}\bar{L}^T$ factorization of $A + \sigma \underline{v}\underline{v}^T$"

  $\Rightarrow$ This is basically a problem of updating LDL^T factorizations of a rank-one corrected matrix

- Problem: updating LDL^T factorizations of a rank-one corrected matrix
  – Starting with

$$A = \sum_{i=1}^{n} d_i \underline{l}_i \underline{l}_i^T \text{ with } \underline{l}_i = \begin{bmatrix} 0 \\ 0 \\ 1 \\ l_{i+1,i} \\ l_{n,i} \end{bmatrix} \begin{matrix} \\ \\ \leftarrow l_{ii} \\ \\ \\ \end{matrix}$$

we want to obtain factorization of $A + \sigma \underline{v}\underline{v}^T = \sum_{i=1}^{n} \bar{d}_i \bar{\underline{l}}_i \bar{\underline{l}}_i^T$

$$\text{i.e.,} \begin{pmatrix} \underline{l}_i \\ d_i \end{pmatrix} \rightarrow \begin{pmatrix} \bar{\underline{l}}_i \\ \bar{d}_i \end{pmatrix}$$

Four Cases for LDL^T Updates

- Consider 4 cases: From general → specific to RLS

1) $\sigma > 0$ arbitrary

2) $\sigma < 0; \sigma = -1/\alpha;\ \alpha > 0$ arbitrary

3) $A = Diag \Rightarrow L = I$

4) $\sigma = -1/\alpha;\ \alpha = 1 + \underline{f}^T D \underline{f};\ \underline{f} = L^T \underline{a}$; special α

Case 1: $\sigma > 0$ arbitrary

- Develop algorithm one column at a time
- Let $\sigma_1 = \sigma; \underline{v}_1 = \underline{v}$. Then

$$\sum_{i=1}^{n} \underline{l}_i \underline{l}_i^T d_i + \sigma_1 \underline{v}_1 \underline{v}_1^T = \underline{l}_1 \underline{l}_1^T d_1 + \sigma_1 \underline{v}_1 \underline{v}_1^T + \sum_{i=2}^{n} d_i \underline{l}_i \underline{l}_i^T$$

- *Q*: can we write

$$\underline{l}_1 \underline{l}_1^T d_1 + \sigma_1 \underline{v}_1 \underline{v}_1^T = \bar{\underline{l}}_1 \bar{\underline{l}}_1^T \bar{d}_1 + \sigma_2 \underline{v}_2 \underline{v}_2^T \qquad (*)$$

where

$$\underline{l}_1 = \begin{bmatrix} 1 \\ * \\ * \\ \cdot \\ \cdot \\ * \end{bmatrix} \quad \text{and} \quad \underline{v}_2 = \begin{bmatrix} 0 \\ \times \\ \times \\ \cdot \\ \cdot \\ \times \end{bmatrix} \Rightarrow \text{first component of } \underline{v}_2, v_{21} = 0.$$

Case 1 (contd.)

- If so, we can generate a recursive scheme to update $\begin{pmatrix} \underline{l}_i \\ d_i \end{pmatrix} \rightarrow \begin{pmatrix} \bar{\underline{l}}_i \\ \bar{d}_i \end{pmatrix}$
- Matrix on LHS has at most rank 2 and range space spanned by $\underline{l}_1$ and $\underline{v}_1$
  $\Rightarrow \underline{l}_1$ and $\underline{v}_2$ must be linear combinations of $\underline{l}_1$ and $\underline{v}_1$
- Let

$$\underline{v}_2 = \underline{v}_1 - v_{11}\underline{l}_1 \text{ since } \underline{v}_{21} = 0 \text{ and } l_{11} = 1$$

$$\bar{\underline{l}}_1 = \underline{l}_1 + \beta_1 \underline{v}_2 = (1-\beta_1 v_{11})\underline{l}_1 + \beta_1 \underline{v}_1$$

where β_1 is arbitrary and is to be determined

- Must solve for "unknowns" $\bar{d}_1, \sigma_2$ and $\beta_1 \ni$ Eqn.$(*)$ is satisfied
- Need 3 eqns. Since we have 3 unknowns: $\bar{d}_1, \sigma_2$ and β_1
- Substitute for v_2 and $\bar{\underline{l}}_1$ into * and equate coefficients.

$$d_1\underline{l}_1\underline{l}_1^T + \sigma_1\underline{v}_1\underline{v}_1^T = \left[(1-\beta_1 v_{11})\underline{l}_1 + \beta_1\underline{v}_1\right]\bar{d}_1\left[(1-\beta_1 v_{11})\underline{l}_1 + \beta_1\underline{v}_1\right]^T$$
$$+\sigma_2\left(\underline{v}_1 - v_{11}\underline{l}_1\right)\left(\underline{v}_1 - v_{11}\underline{l}_1\right)^T$$

Case 1 (contd.)

coeff. of $\underline{v}_1\underline{v}_1^T : \sigma_1 = \bar{d}_1\beta_1^2 + \sigma_2 \Rightarrow \sigma_2 = \sigma_1 - \bar{d}_1\beta_1^2$

coeff. of $\underline{l}_1\underline{v}_1^T : 0 = 2\bar{d}_1\left(1-\beta_1 v_{11}\right)\beta_1 - 2\sigma_2 v_{11} = 0 \Rightarrow \beta_1 = \sigma_1 v_{11} / \bar{d}_1$

coeff. of $\underline{l}_1\underline{l}_1^T : d_1 = \left(1-\beta_1 v_{11}\right)^2 \bar{d}_1 + \sigma_2 v_{11}^2$

$$\begin{aligned} &= \left(1-\beta_1 v_{11}\right)^2 \bar{d}_1 + \sigma_1 v_{11}^2 - \bar{d}_1 v_{11}^2 \beta_1^2 \\ &= \bar{d}_1 - 2\beta_1 v_{11}\bar{d}_1 + \sigma_1 v_{11}^2 \\ &= \bar{d}_1 - 2\sigma_1 v_{11}^2 + \sigma_1 v_{11}^2 \end{aligned}$$

$\Rightarrow \; \bar{d}_1 = d_1 + \sigma_1 v_{11}^2$. Also, note $\sigma_2 = \sigma_1 - \bar{d}_1\beta_1^2 = \sigma_1\left(1 - \frac{\sigma_1 v_{11}^2}{\bar{d}_1}\right) = \sigma_1 \frac{d_1}{\bar{d}_1}$

$\Rightarrow$ So, we compute $\bar{d}_1, \sigma_2$ and β_1 in that order

- Next, repeat with $\underline{l}_2\underline{l}_2^T d_2 + \sigma_2\underline{v}_2\underline{v}_2^T \rightarrow \bar{\underline{l}}_2\bar{\underline{l}}_2^T d_2 + \sigma_3\underline{v}_3\underline{v}_3^T$

  where $v_{31} = v_{32} = 0$

Update Algorithm for Case 1

Initialize $\sigma_1 = \sigma; \underline{v}_1 = \underline{v}$

For $k = 1, 2, \ldots, n-1$ DO

1) $\bar{d}_k = d_k + \sigma_k v_{kk}^2$

2) $\beta_k = \sigma_k v_{kk} / \bar{d}_k$

3) $\sigma_{k+1} = \sigma_k \dfrac{d_k}{\bar{d}_k}$

4) $\underline{v}_{k+1} = \underline{v}_k - v_{kk} \underline{l}_k$ note: $\underline{v}_{k+1} = \begin{bmatrix} 0 \\ . \\ 0 \\ \times \\ \times \end{bmatrix} \leftarrow k.$

So, we need to compute elements $(k+1, \ldots n)$ only

5) $\underline{\bar{l}}_k = \underline{l}_k + \beta_k \underline{v}_{k+1}$

End DO

$\bar{d}_n = d_n + \sigma_n v_{nn}^2$

Case 2: $\sigma < 0$ (as in RLS)

- In this case, we may end up in a situation where $\bar{d}_k < 0$ in step 1.
- This may be due to round-off or near rank degeneracy of P_k
- Need slightly different formulae:

Let $\sigma_k = -1/\alpha_k$; assume $\alpha_k > 0$ with $\alpha_1 = \alpha = -1/\sigma$

- Step 3 of algorithm implies

$$\bar{d}_k = \frac{\alpha_{k+1}}{\alpha_k} d_k \quad \text{(step 3a)}$$

maintains positivity of $\bar{d}_k$ if $d_k > 0$

- Substitute in step 1: $\bar{d}_k = d_k + \sigma_k v_{kk}^2 = d_k - \dfrac{v_{kk}^2}{\alpha_k}$

$$\frac{\alpha_{k+1}}{\alpha_k} d_k = d_k - \frac{v_{kk}^2}{\alpha_k}$$

$$\Rightarrow \alpha_{k+1} = \alpha_k - \frac{v_{kk}^2}{d_k}$$

Case 2 (contd.)

or $\alpha_k = \alpha_{k+1} + \dfrac{v_{kk}^2}{d_k}$ (Step 1a)

α_k^S are postive provided that α_k^S are computed backwards. Need α_{n+1} to initialize recursion and all v_{kk}

It is easy to initialize α_{n+1} in RLS (e.g., $\alpha_{n+1} = 1$ or c)

$$\text{if } \alpha = \underline{f}^T D \underline{f} + c \Rightarrow \alpha_{n+1} = c$$

- Also, have

$$\beta_k = -\frac{v_{kk}}{\alpha_k \bar{d}_k} = -\frac{v_{kk}}{\alpha_{k+1} d_k} \qquad \text{(Step 2a)}$$

- Key: when A is a diagonal matrix as in case 3, we can compute v_{kk} *a priori*

Case 3 : A = Diag (d_i)

- $A = Diag(d_i) \Rightarrow \underline{l}_i = \underline{e}_i =$ unit vectors
  - Note that $\underline{v}_{k+1} = \underline{v}_k - v_{kk}\underline{e}_k$
    $= \underline{v}_k$ with k^{th} element set to 0
  - So, if $$\underline{v} = \underline{v}_1 = \begin{bmatrix} v_1 \\ v_2 \\ . \\ . \\ v_n \end{bmatrix}, \text{ then } \underline{v}_k = \begin{bmatrix} 0 \\ . \\ v_k \\ . \\ v_n \end{bmatrix} = \underline{v}_k^{(k)}; \left(\underline{v}^{(1)} = \underline{v}; \underline{v}^{(n+1)} = \underline{0}\right)$$
  - Thus, the v_{kk} in the above case are known *a priori*

    $$\Rightarrow \underline{\bar{l}}_k = \underline{e}_k + \beta_k \underline{v}^{(k+1)} \text{ or } \bar{l}_{ik} = \beta_k v_i;\ i > k$$

Case 4 : Finally *RLS*

- Special case for RLS $\underline{v} = D\underline{f}$; $\underline{f} = L^T\underline{a}$ arbitrary since $\underline{a}$ is arbitrary
  - $\alpha = \underline{f}^T D\underline{f} + c;\ c = \text{scalar}$
  - So, here $\alpha = c + \sum_{i=1}^{n} f_i^2 d_i = \alpha_1$ and $v_k = v_{kk} = d_k f_k$
  - But from (1a): $\alpha_k = \alpha_{k+1} + f_k^2 d_k$

    $\Rightarrow \alpha_{n+1} = c$

    $\Rightarrow$ can get α_k via a backward recursion.
  - So, if $c \geq 0, \alpha_k \geq 0 \Rightarrow \bar{d}_k > 0$ (see step 3a)
  - Also, note $\beta_k = -v_k / (d_k \alpha_{k+1}) = -f_k / \alpha_{k+1}$
  - Since all $\alpha_k, \bar{d}_k$ and β_k can be computed *a priori*, we can get L (i.e., $\underline{\bar{l}}_i$) either forward or backward. But backward is preferred, since we don't have to store α_k and β_k.

Algorithm for RLS

- Due to Agee-Turner (1972); Gill, Golub, Murray & Saunders (1974)

Initialize $\alpha_{n+1} = c$

$\alpha_n = c + f_n^2 d_n$

$\bar{d}_n = \left(\alpha_{n+1} / \alpha_n\right) d_n$

For $k = n-1,\ldots,1$ DO

$$\beta_k = -f_k / \alpha_{k+1} \qquad (2a)$$

$$\alpha_k = \alpha_{k+1} + d_k f_k^2 \qquad (1a)$$

$$\bar{d}_k = d_k . \alpha_{k+1} / \alpha_k \qquad (3a)$$

$$\bar{\underline{l}}_k = \underline{l}_k + \beta_k \underline{v}^{(k+1)}$$

end DO

- Once done, we have

$$\tilde{L} = I + \begin{bmatrix} | & | & .. & | & | \\ \beta_1 \underline{v}^{(2)} & \beta_2 \underline{v}^{(3)} & .. & \beta_{n-1} \underline{v}^{(n)} & 0 \\ | & | & .. & | & | \end{bmatrix}$$

Algorithm for RLS – Details - 1

- For Least Squares, we need

$$L\left[D-\underline{v}\,\underline{v}^T/\left(c+\underline{f}^T D\underline{f}\right)\right]L^T=\bar{L}\bar{D}\bar{L}^T \quad \text{where } \underline{v}=D\underline{f}$$

$$\text{so factor }\left[D-\underline{v}\,\underline{v}^T/\left(c+\underline{f}^T D\underline{f}\right)\right]\rightarrow \tilde{L}_1\tilde{D}_1\tilde{L}_1^T \Rightarrow \bar{L}=L\tilde{L}_1$$

- But can get $\bar{L}$ directly. From above multiply by L:

$$\bar{L}=L+\begin{bmatrix} | & | & .. & | & | \\ \beta_1 L\underline{v}^{(2)} & \beta_2 L\underline{v}^{(3)} & .. & \beta_{n-1}L\underline{v}^{(n)} & 0 \\ | & | & .. & | & | \end{bmatrix}$$

$$\text{Define } L\underline{v}^{(k)}=\underline{\xi}_k,\ \text{since } l_{ii}=1\Rightarrow \underline{\xi}_k=\begin{bmatrix}0\\ \cdot\\ v_{kk}\\ \times\\ \times\end{bmatrix};\underline{\xi}_n=\begin{bmatrix}0 & \dots & 0 & v_n\end{bmatrix}$$

Algorithm for RLS - Details - 2

$$\Rightarrow \text{So, } \underline{\tilde{l}}_k = \underline{l}_k + \beta_k \underline{\xi}_{k+1}$$

$$\Rightarrow \underline{\xi}_k = L\underline{v}^{(k)} = L\underline{v}^{(k+1)} + \underline{l}_k v_{kk} \text{ since } \underline{v}^{(k+1)} = \underline{v}^k - \underline{e}_k v_{kk}$$

$$\Rightarrow \underline{\xi}_k = \underline{\xi}_{k+1} + v_{kk}\underline{l}_k = \left(1 - \beta_k v_{kk}\right)\underline{\xi}_{k+1} + \underline{\tilde{l}}_k v_{kk}$$

- Note 1: $\underline{\xi}_1 = L\underline{v}_1$

  the gain vector, $\underline{g} = P\underline{a} / \left(c + \underline{a}^T P\underline{a}\right)$

  $$= LDL^T\underline{a} / \left(f^T Df + c\right) = \underline{\xi}_1 / \alpha_1$$

  $$\Rightarrow \boxed{\underline{g} = \frac{1}{\alpha_1} \cdot \underline{\xi}_1}$$

- Note 2: Start the entire process with $L = I$, $D = 10^5 I$
  Computational load = O $(1.5n^2 + 2.5n)$

Sequential QR – Add a Measurement - 1

- Adding a new measurement $\Rightarrow$ add a new row to A
  - Suppose added new row as row 1

$$A_{k+1} = \begin{bmatrix} a_{k+1}^T \\ A_k \end{bmatrix} \text{ where } A_k = Q_k R_k,\ Q_k \to k \times n \text{ and } R_k \to n \times n$$

$$diag\left(1, Q_k^T\right) A_{k+1} = \begin{bmatrix} a_{k+1}^T \\ R_k \end{bmatrix} = H_k \Rightarrow \text{ Upper Hessenberg matrix}$$

  - For k = 4 and n = 3, H_k looks like:

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & 0 \end{bmatrix}$$

  - H_k = upper Hessenberg (upper Δ + sub diagonal)

Sequential QR – Add a Measurement - 2

- Apply Givens transformations to upper triangularize H_k

$$J^T(n,n+1)\ldots J^T(2,3)J^T(1,2)H_k = R_{k+1}$$

$$\Rightarrow J_n^T \ldots J_2^T J_1^T H_k = R_{k+1}$$

$$\Rightarrow A_{k+1} = Q_{k+1}R_{k+1}$$

where $Q_{k+1} = diag(1,Q)J_1J_2\ldots J_n$

- But want to add a row k+1 at the end $\quad A_{k+1} = \begin{bmatrix} A_k \\ \underline{a}_{k+1}^T \end{bmatrix}$

- Use exchange matrix $\quad E_k = \begin{bmatrix} 0 & . & \ldots & 1 \\ 0 & .. & 1 & 0 \\ 1 & .. & . & 0 \end{bmatrix}$ k by k matrix; $E_k^2 = I_k$

and define $\bar{A} = \begin{bmatrix} \underline{a}_{k+1}^T \\ E_k A_k \end{bmatrix}, \bar{Q}_k = E_k Q_k$

$$\Rightarrow diag(1,\bar{Q}_k^T)\bar{A} = \begin{bmatrix} \underline{a}_{k+1}^T \\ R_k \end{bmatrix} = H_k$$

Sequential QR – Add a Measurement - 3

- So apply Givens transformations as before to obtain:

$$J_n^T \ldots J_2^T J_1^T H_k = R_{k+1}$$

$$J_n^T \ldots J_2^T J_1^T \underbrace{diag\left(1, \bar{Q}_k^T\right) E_{k+1} A_{k+1}}_{\bar{A}} = R_{k+1}$$

$$\Rightarrow Q_{k+1} = E_{k+1} diag\left(1, \bar{Q}_k^T\right) J_1 \ldots J_n$$

$$= \begin{bmatrix} 0 & E_k \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & E_k Q_k \end{bmatrix} J_1 \ldots J_n$$

$$= \begin{bmatrix} 0 & Q_k \\ 1 & 0 \end{bmatrix} J_1 \ldots J_n$$

- O(m n) operations

Sequential QR – Drop a Measurement

- Suppose we want to delete a measurement (e.g., found to be an outlier after it was incorporated into Least Squares estimate)
  - For simplicity, assume it is 1st measurement

$$A_k = \begin{bmatrix} \underline{a}_1^T \\ A_1 \end{bmatrix}$$

  - Let $\underline{q}_1^T$ be the first row of Q
  - Compute Givens rotations $J_{m-1} \ldots J_1$

$$J_{m-1} \ldots J_1 \underline{q}_1 = \alpha \underline{e}_1 \text{ where } \alpha = \pm 1$$

$$H = J_1^T \ldots J_{m-1}^T R = \begin{bmatrix} \underline{v}^T \\ R_1 \end{bmatrix},$$

where $H =$ upper Hessenberg matrix

  - Note that $QJ_{m-1} \ldots J_1 = \begin{bmatrix} \alpha & 0 \\ 0 & Q_1 \end{bmatrix}$

Sequential QR – Add a Parameter

- It must be of this form, since it is orthogonal
- So, $A = \begin{bmatrix} \underline{a}_1^T \\ A_1 \end{bmatrix} = \left(QJ_{m-1} \ldots J_1\right) J_1^T \ldots J_{m-1}^T R$

$$= \begin{bmatrix} \alpha & 0 \\ 0 & Q_1 \end{bmatrix} \begin{bmatrix} \underline{v}^T \\ R_1 \end{bmatrix}$$

$\Rightarrow A_1 = Q_1 R_1$ is the desired Q-R factorization

- ❑ Adding a new column => increase number of parameters by 1

$$\bar{A} = \begin{bmatrix} \underline{a}_1 & \underline{a}_2 & \ldots & \underline{a}_n & \underline{a}_{n+1} \end{bmatrix}$$

- So, $Q^T \bar{A} = \begin{bmatrix} R\underline{w} \end{bmatrix}; \underline{w} = Q^T \underline{a}_{n+1}$

Sequential QR – Drop a Parameter - 1

- Apply $J_{n+1}^T \ldots J_{m-1}^T w = \begin{bmatrix} \times \\ \times \\ \times \\ 0 \\ . \\ 0 \end{bmatrix} \leftarrow n+1 \quad \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$

  $\Rightarrow$ does not change R and $Q = QJ_{m-1} \ldots J_{n+1}$

  $\Rightarrow$ computational load: O(mn) operations

❑ Delete column $k \Rightarrow$ remove x_k (or k^{th} factor) $\Rightarrow$ reduce the number of parameters by 1

- $\bar{A} = [\underline{a}_1 \quad \ldots \quad \underline{a}_{k-1} \quad \underline{a}_{k+1} \quad \ldots \quad \underline{a}_n]$

Sequential QR – Drop a Parameter - 2

- Write

$$Q^T A = \begin{array}{c|c} \begin{array}{ccc} k-1 & 1 & n-k \end{array} & \\ \begin{bmatrix} R_{11} & v & R_{12} \\ 0 & r_{kk} & \underline{w}^T \\ 0 & 0 & R_{33} \end{bmatrix} & \begin{array}{c} k-1 \\ 1 \\ m-k \end{array} \end{array}$$

$$Q^T \bar{A} = \begin{bmatrix} R_{11} & R_{12} \\ 0 & \underline{w}^T \\ 0 & R_{33} \end{bmatrix} \begin{array}{c} k-1 \\ 1 \\ m-k \end{array} = H \Rightarrow \text{Upper Hessenberg from columns } (k+1) \text{ to } n$$

- Consider m = 5, n = 4, k = 2

$$\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{bmatrix}$$

$$\Rightarrow J_{n-1}^T \ldots J_{k+1}^T J_k^T H = R_1$$

$$\Rightarrow Q = Q J_k J_{k+1} \ldots J_{n-1}$$

Computational load: O(n^2)

- Zero out unwanted sub diagonals $h_{k+1,k} \ldots h_{n,n-1}$

Application to Kalman Filtering - 1

- Consider the LTI dynamic model of a stochastic system:

Dynamics: $\underline{x}_{k+1} = \Phi \underline{x}_k + E \underline{w}_k$

Measurement: $\underline{y}_k = H \underline{x}_k + \underline{v}_k$

- Note that Φ, G, and H can be time-varying
- But, we will assume that they are time-invariant for simplicity of notation
- $\{\underline{w}_k\}$ process noise sequence. Assumed to be zero-mean white, Gaussian noise sequence with covariance matrix, W_d
- $\{\underline{v}_k\}$ measurement noise sequence. Assumed to be zero-mean, white Gaussian noise sequence with covariance matrix R
- Without loss of generality, assume that W_d and R are diagonal

Application to Kalman Filtering - 2

- **If not:**

Form $W_d = L_w D_w L_w^T$

$$\text{and define:}\quad \left.\begin{aligned} &\text{new } E = E L_w \\ &\text{new } W_d = D_w \\ &\text{new } \underline{w}_k = L_w^{-1}\underline{w}_k \end{aligned}\right\} \text{called whitening of process noise}$$

- **Similarly, if R is not diagonal**

Form $R = L_r D_r L_r^T$

$$\text{and define:}\quad \left.\begin{aligned} &\text{new } \underline{y}_k = L_r^{-1}\underline{y}_k \\ &\text{new } H = L_r^{-1} H \\ &\text{new } R = D_r \\ &\text{new } \underline{v}_k = L_r^{-1}\underline{v}_k \end{aligned}\right\} \text{called whitening of observation errors}$$

- Kalman filter provides the minimum mean-square error (MMSE) estimate (also called maximum a posteriori (MAP) estimate).
- If the initial state $\underline{x}_0$ is Gaussian with mean $\underline{\bar{x}}_0$ and covariance matrix P_0, define:
- $\hat{\underline{x}}_{k/k}$ = best estimate of $\underline{x}_k$ based on measurements $\{\underline{y}_1, \ldots \underline{y}_k\}$
- $\hat{\underline{x}}_{k+1/k}$ = best estimate of $\underline{x}_{k+1}$ based on measurements $\{\underline{y}_1, \ldots \underline{y}_k\}$

Kalman Filter Equations

- The Kalman filter equations are:

$$\underline{\hat{x}}_{k+1/k} = \Phi \underline{\hat{x}}_{k/k} \quad \text{.... (PROPAGATE or PREDICTION STEP)}$$

$$\underline{\hat{x}}_{k/k} = \underline{\hat{x}}_{k/k-1} + \underbrace{G_k}_{\substack{\text{Kalman}\\ \text{Gain}}} \underbrace{(y_k - H\underline{\hat{x}}_{k/k-1})}_{\text{innovation}} \quad \text{.... (UPDATE STEP)}$$

- Different filter algorithms differ in the way they compute the Kalman gains $\{G_k\}$
  - Conventional Kalman filter:

$$G_k = P_{k/k-1} H^T \left(H P_{k/k-1} H^T + R \right)^{-1}$$

– update step: $P_{k/k} = (1 - G_k H) P_{k/k}$

– propagate step: $P_{k+1/k} = \Phi P_{k/k} \Phi^T + E W_d E^T$

where $P_{k/k} = E\left[\left(\underline{x}_k - \underline{\hat{x}}_{k/k} \right) \left(\underline{x}_k - \underline{\hat{x}}_{k/k} \right)^T \right]$

$$P_{k+1/k} = E\left[\left(\underline{x}_{k+1} - \underline{\hat{x}}_{k+1/k} \right) \left(\underline{x}_{k+1} - \underline{\hat{x}}_{k+1/k} \right)^T \right]$$

Round-off Error Problems

Remarks on the conventional Kalman filter:

- The update step can be implemented recursively one measurement at a time. This is because:

$$P_{k/k}^{-1} = P_{k/k-1}^{-1} + H^T R^{-1} H = P_{k/k-1}^{-1} + \sum_{i=1}^{m} r_i \underline{h}_i \underline{h}_i^T$$

$$\underline{h}_i^T = i^{th} \text{ row of } H$$

$$r_i = i^{th} \text{diagonal element of } R$$

$$m = \text{number of measurements} = \#\text{of rows in } H$$

- Need to compute only $n(n+1)/2$ elements of P, since P is symmetric.
- Could get negative diagonal elements in P $\left(\text{if } r_i \text{ and/or } w_{di} \text{ are small}\right)$.

Joseph's stabilized measurement update:

$$P_{k/k} = \left(1 - G_k H\right) P_{k/k-1} \left(1 - G_k H\right)^T + G_k R G_k^T$$

$$= \left(1 - G_k H\right) P_{k/k-1} \left(1 - G_k H\right)^T + \sum_{i=1}^{m} r_i \underline{g}_{ki} \underline{g}_{ki}^T, \quad \underline{g}_{ki} = i^{th} \text{column of } G_k$$

Solution Approaches

❑ Conventional Kalman filter with lower bounding:

- Compute:

$$\bar{P} = (1 - G_k H) P_{k/k-1}$$

$$(P_{k/k})_{jj} = \max\left(\bar{P}_{jj}, \sigma^2_{\min,j}\right); \quad j = 1, 2, \ldots, n$$

$$(P_{k/k})_{ij} = \begin{cases} \bar{P}_{ij} \text{ if } \bar{P}^2_{ij} < M_{ij} \\ \text{sign}\left(\bar{P}_{ij}\right)\sqrt{M_{ij}} \text{ otherwise} \end{cases}$$

$$\text{where } M_{ij} = \rho^2_{\min} (P_{k/k})_{ii} (P_{k/k})_{jj}$$

– Selection of $\rho_{\min}$ and $\sigma_{\min}$ is an art

– Does not guarantee positive definiteness of $P_{k/k}$

(see Kerr, IEEE T-AES, Nov. 1990)

❑ LDL^T Factorization:

- We will present the **algorithm in two steps:**

1. update step
2. propagation step

Kalman Filter via LDL^T Updates

❑ LDL^T Factorization: Measurement update step

- **Trivial application of previous Least-squares update algorithm**
  - We know that $P_{k/k}^{-1} = P_{k/k-1}^{-1} + \sum_{i=1}^{m} r_i \underline{h}_i \underline{h}_i^T$
  - So, implement via:

    DO $i = 1, m$

    call previous Agee-Tumer algorithm with $(r_i, \underline{h}_i)$ and current L and $\underline{d}$

    end DO

- **Factorization problem associated with propagation step:**
  - Recall that: $P_{k+1/k} = \Phi P_{k/k} \Phi^T + E W_d E^T$
  - Problem: Given $P_{k/k} = LDL^T$, we seek $\bar{L}, \bar{D}$ such that $P_{k+1/k} = \bar{L}\bar{D}\bar{L}^T$

LDL^T Update Methods for Kalman Filters

- There are basically 3 methods to obtain $\bar{L}, \bar{D}$ of $P_{k+1/k}$:
  - **Method 1**
    - Let $EW_dE^T = L_eD_eL_e^T$ be the factorization of EW_dE^T
    - Further, let $\underline{l}_i = \underline{\gamma}_i$ for $i = 1, 2, \ldots, n,$ where $\underline{l}_i =$ column i of L
    - Then:

$$P_{k+1/k} = L_eD_eL_e^T + \underbrace{\sum_{i=1}^{m} d_i \underline{\gamma}_i \underline{\gamma}_i^T}_{\text{modifying rank-one corrections}}$$

    - So, the algorithm is:

      DO $i = 1, n$

      call case 1 of general algorithm with $\left(d_i, \underline{\gamma}_i\right)$

      end DO

    - Probem: EW_dE^T is typically positive semi-definite

Gram-Schmidt for Propagation Step - 1

Method 2: Weighted Gram-Schmidt

- Recall that $P_{k+1/k} = [\Phi L \quad E]\begin{bmatrix} D & 0 \\ 0 & W_d \end{bmatrix}[\Phi L \quad E]^T = A^T \tilde{D} A = \bar{L}\bar{D}\bar{L}^T$

  where $A = \begin{bmatrix} L^T \Phi^T \\ E^T \end{bmatrix}$, $\tilde{D} = \begin{bmatrix} D & 0 \\ 0 & W_d \end{bmatrix}$

  – Obtain $\bar{L}\bar{D}\bar{L}^T$ via parallel weighted Gram-Schmidt orthogonalization procedure.

  – Idea: Obtain a set of orthogonal directions $(\underline{q}_1, \underline{q}_2, ..., \underline{q}_n)$ where $\underline{q}_i \in R^{n+n_w}$

  $$A = \begin{bmatrix} L^T \Phi^T \\ E^T \end{bmatrix} = (\underline{a}_1, \underline{a}_2, ..., \underline{a}_n) = [\underline{q}_1, \underline{q}_2, ..., \underline{q}_n]\bar{L}^T = Q\bar{L}^T = QR$$

  where $\bar{L}$ = unit lower Δ, R = unit upper Δ

  and $\underline{q}_i^T \tilde{D} \underline{q}_j = 0 \ \forall \ i \neq j \ \Rightarrow \ \{\underline{q}_i\}$ are $\tilde{D}$-orthogonal.

  – Once $\underline{q}_i^T$ are known, we can obtain $\bar{D} = diag(\bar{d}_1, \bar{d}_2, ..., \bar{d}_n)$ via:

  $$\bar{d}_i = \underline{q}_i^T \tilde{D} \underline{q}_i; \ i = 1, 2, .., n$$

  – After D-orthogonalization, $P_{k+1/k} = A^T \tilde{D} A = \bar{L} Q^T \tilde{D} Q \bar{L}^T = \bar{L}\bar{D}\bar{L}^T$

Gram-Schmidt for Propagation Step - 2

Method 2 cont…

- The algorithm for $\tilde{D}$-orthogonalization of A is a minor variation of parallel Gram-Schmidt.
  - The matrix A is replaced by Q
  - Algorithm:

For $k = 1, 2, \ldots, n$ DO

$$\bar{d}_k = \underline{a}_k^T \tilde{D} \underline{a}_k$$

$$r_{kk} = 1 \quad \rightarrow \; l_{kk} = 1$$

For $j = k+1, \ldots, n$ DO

$$r_{kj} = \frac{\underline{a}_j^T \tilde{D} \underline{a}_k}{\bar{d}_k} \quad \rightarrow \text{ obtained } l_{jk}$$

$$\underline{a}_j \leftarrow \underline{a}_j - r_{kj} \underline{a}_k$$

end DO

end DO

Method 3: Householder or Givens Transformations

- Recall that we can find transformation Q such that

$$\tilde{D}^{1/2} A = Q \bar{D}^{1/2} \bar{L}^T$$

$$\rightarrow P_{k+1/k} = \mathrm{A}^T \tilde{\mathrm{D}} \mathrm{A} = \bar{\mathrm{L}} \bar{\mathrm{D}} \bar{\mathrm{L}}^T$$

Other Applications of Sqrt Updates

- **Other applications of square-root updates**
  - Probabilistic data association filter (PDAF) to track targets in clutter – additional m rank-one corrections in the measurement update equations
  - Quasi-Newton methods in non-linear programming
    - rank-two or rank-three corrections

- **References:**

1.) G.J. Bierman, Factorization Methods for Discrete Sequential Estimation, Academic Press, 1977.
2.) P.E. Gill, G.H. Golub, W. Murray and M.A. Saunders, "Methods for Modifying Matrix Factorizations," Mathematics of Computation, Vol. 28, No. 126, April 1974, pp. 505-535.
3.) "Special Issue on Factorized Estimation Applications, IEEE Trans. On Automatic Control, Dec. 1990.
4.) V. Raghavan, K.R. Pattipati and Y. Bar-Shalom, "Efficient L-D Factorization Algorithms for PDA, IMM, and IMM-PDA Filters," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 29, October 1993, pp. 1297-1310.

Summary

- Recursive (sequential) Least Squares
- Sequential LDL^T Factorization updates
- Sequential QR updates
- Application to Kalman filtering