

**Prediction**

We may want to predict \( N \) steps ahead from time \( k \) an estimate of the state variables and the output of a system with process and measurement noise. We will use the following notation for prediction:

\[
\hat{X}(k+N) = \phi(k+N,k) \hat{X}(k) \quad (6.1.1)
\]

That is, we compute for \( N \) steps ahead and multiply it by the current kalman filter estimate of the state. We modify the kalman filter (see Figure 5.8) as shown in figure 6.1

**Figure 8.1** \( N \)-step prediction.

The predicted error covariance, \( P(k+N) \), does not need to be computed. However, computing \( P(k+N) \) provides an estimate of the prediction error. That is, the diagonal of \( \sqrt{P(k+N)} \) is the root-mean-square (RMS) state-variable error.

**Estimation of Prediction Error**

We can estimate the prediction error by initialising \( P \) to all zeros and computing:

\[
\hat{x}(k+N|k) = \phi(k+N,k) \hat{x}(k|k)
\]

\[
P(k+N|k) = \phi(k+N,k) P(k|k) \phi^T(k+N,k) + Q(k+N,k)
\]
The plot of $\text{rms}\_e$ is a plot of the RMS state variable prediction error as a function of the prediction step size. See Example 6.1 and Figure 6.2 in the textbook.

**Alternate Kalman Filter Form**

There are several different kalman filter forms. One form that is useful if $P(0) = \infty$ (infinite uncertainty). In this case, we compute $P_k$ first. Otherwise, if we compute the otherwise, if we compute the gain $K_k$ first, $K_k$ would "Blow up".

However, using this form, as shown in Figure 6.3 below, more computation time is required.
Stability Considerations

The characteristic polynomial of the discrete kalman filter for a discrete LTI system is

\[
\det[\lambda I - (\Phi_{k-1} - K_k H_k \Phi_{k-1})]
\]

If the eigenvalues (Roots) lie inside the unit circle, i.e.,

\[
|\lambda_i| < 1, \ i = 1, 2, \ldots, n
\]

The Kalman Filter is inherently stable providing an appropriate P(0) is chosen and P(k) is maintained as a symmetric and positive definite matrix.

Choice of P(0)

A kalman filter solves the following "Riccati Equation" of LTI systems:
\[ P_k = f_{k-1}H_k^T \left[ H_k f_k - H_k^T + R_k \right]^{-1} H_k f_{k-1} \]

where
\[ f_{k-1} = \phi_{k-1} P_{k-1} \phi_{k-1}^T + \Gamma_{w_{k-1}} \Gamma_{w_{k-1}} \]

Stable solutions are dependant on \( P(0) \) for example.

**Kalman Filtering, Grewal & Andrews, Prentice-Hall, 1993**
Chapter

Hence, selection of an improper initial value can lead to blow-up of a Kalman filter. For a continuous-time LTI system, we can, generally, estimate the initial covariance \( P(0) \) and use it as \( P(\infty) \).

**Practical Considerations**

*** Prefiltering can sometimes help yield better results. Particularly if the system "dynamites" are "slow" as compared to the sampling rate.

*** Detect anomalous sensor data before passing it to the Kalman filter.

*** Test for asymptotic stability/convergence could check the trace of \( P_{cov} \). If trace of \( P_{cov} \) continues to decrease then \( P_{cov} \) is most likely converging.
*** Nonconvergence of $P_{cov}$ can be due to

** Roundoff errors (use double precision, or reduce # states)

** Unstable system (not kalman filter)

** Nonobservability of states

** Bad Data (need bad data detector)

** Mismodeling

* Unmodeled state variables

* Trick kalman filter by adding another state as process noise

* Do a frequency analysis of $z_k - \hat{z}_k$ may indicate an unmodeled state

* Unmodeled process noise

*** Suboptimal Filters

** Reduce model complexity and/or number of states

** Linearized kalman filter

** Extended kalman filter
** Use constant $\hat{\mathbf{k}}$

** Prefiltering

** Use frequency domain to obtain a linear model

** Engineering insight. Observe eigenvalues (poles)

** Compare suboptimal simulations with "Full" model simulations

### Implementation Considerations

**** Kalman filter iteratively solves the "Riccati Equation". Round-off and finite precision can result in $P_{cov}$ variances that are negative - A theoretical impossibility. Force $P_{cov}$ variances to be greater than zero, check the Kalman filter state estimates for reasonableness, check the input to the kalman filter and report that an error condition has occurred.

**** If it is an ill-conditioned problem, then the solution is too sensitive to input data. Example, matrix, $P$, is close to being singular - a good measure is $\det[P]$. In this case "Tweak" the model or use a suboptimal approach.

Condition number of a matrix, $\text{cond}(P)$ indicates how close the $\det[P]$ is to zero.

$$ 1 \leq \text{cond}(P) \leq \infty $$

$$ \text{cond}(P) = \| P^{-1} \| \| P \| $$

or

$$ \text{cond}(P) = \| P^+ \| \| P \| $$

if A is singular or nonsquare ($P^+$ is the Pseudo Inverse)

or

$$ \text{cond}(P) = \frac{\max_j |\lambda_j(P)|}{\max_i |\lambda_i(P)|} $$

if square.

**** Of factorization methods Bierman UD factorization is one of the more stable and efficient algorithms to keep $P_{cov}$ symmetric.

$$ P_{cov} = UDU^T $$

referred to as a "Square Root Filter", but is not used.

\[
\mathbf{P} = \left[ \mathbf{U} \sqrt{\mathbf{D}} \right] \left[ \mathbf{U} \sqrt{\mathbf{D}} \right]^T
\]

However, which is why it is wrongly referred to as a "Square Root Factorization".

\[ \mathbf{P}_{\text{cov}} = \mathbf{U} \mathbf{D} \mathbf{U}^T \]

is not likely to be indefinite even in the presence of round-off error, by updating only \( U \) and \( D \).

**Kalman Filtering, Grewal & Andrews, Prentice-Hall, 1993**

*Partial UD factorization of the covariance equations.* In a manner similar to the case with Cholesky factors for scalar-valued measurements, the conventional form of the observational update of the covariance matrix:

\[
P(+) = P(-) - \frac{P(-) \mathbf{H}^T \mathbf{H} P(-)}{\mathbf{R} + \mathbf{H} P(-) \mathbf{H}^T}
\]

can be partially factored in terms of \( UD \) factors:

\[
P(-) \overset{\text{def}}{=} U(-) D(-) U^T(-) \quad (6.94)
\]

\[
P(+) \overset{\text{def}}{=} U(+) D(+) U^T(+) \quad (6.95)
\]

\[
U(+) D(+) U^T(+) = U(-) D(-) U^T(-)
\]

\[
\begin{align*}
&\quad \quad \frac{U(-) D(-) U^T(-) \mathbf{H} \mathbf{U}(-) D(-) U^T(-)}{\mathbf{R} + \mathbf{H} U(-) D(-) U^T(-) \mathbf{H}^T}
\end{align*}
\]

\[
= \quad U(-) D(-) U^T(-) - \frac{U(-) D(-) \mathbf{v} \mathbf{v}^T D(-) U^T(-)}{\mathbf{R} + \mathbf{v}^T D(-) \mathbf{v}}
\]

\[
= \quad U(-) \left[ D(-) - \frac{D(-) \mathbf{v} \mathbf{v}^T D(-)}{\mathbf{R} + \mathbf{v}^T D(-) \mathbf{v}} \right] U^T(-)
\]

(6.97)

(6.98)

where

\[
\mathbf{v} = U^T(-) \mathbf{H}^T
\]

(6.99)

is an \( n \)-vector, and \( n \) is the dimension of the state vector.
If roundoff errors or ill-conditioned covariance matrix produce an indefinite or negative definite matrix or nonsymmetric matrix, we can use Joesph's Method then:

1) Force $P_{\text{cov}}$ to be symmetric:

$$P_{\text{cov}} = \frac{1}{2} (P_{\text{cov}} + P_{\text{cov}}^T)$$

2) Then use eigenvalue-eigenvector decomposition (Matlab eig.m):

$$P_{\text{cov}} = T \Lambda T^T$$

\[\text{D} = \text{DIAGONAL OF EIGENVALUES}\]
\[\text{T} = \text{COLUMNS OF EIGENVECTORS}\]

For $i = 1:M$  
% REPAIR $\Delta$

if $\delta_{ii} < 0$
  $\% \delta_{ii} \in D_k$
endif

if $\delta_{ii} = 0$
  $\% \delta_{ii} \in \Phi_k$
endif

\[\delta_{ii} = 0;\]

else

\[\delta_{ii} = \epsilon; \% \epsilon > 0\]
3) MDH method:

Force upper-left submatrices (Lenoing Principal Minors) determinants to be \( \geq 0 \). Distributes changes along last column and row of submatrix being evaluated.

**EXAMPLE:**

\[
D = \begin{bmatrix}
    p_{11} & p_{12} \\
    p_{21} & p_{22}
\end{bmatrix};
\]

\( p_{22} = p_{12} \)

**Upper-left submatrices:**

For \( i = 1 : M \),

\[
\text{if } (p_{ii} \leq 0) \quad \text{Repair diagonal}
\]

\[
\text{if } (p_{ij} = 0) \quad p_{ij} = 0
\]

\[
\text{if } (p_{ii} = 0) \quad p_{ii} = 0
\]
else

\[ \hat{p}_{ii} = \varepsilon \]

end

end

end

if \( P_{12} = 0 \)

\[ P_{12} = 0; \hat{p}_{21} = 0; \]

else

if \( \text{det}(P) < 0 \)

\[ P_{12} = P_{12}^*; \hat{p}_{21} = \hat{p}_{21}^*; \]

else

\[ \text{det} [P_{11} P_{22}] = P_{11} P_{22} - P_{12}^2 \]

\[ \geq \sqrt{P_{11} P_{22} - \varepsilon} \]

\[ P_{12} = \sqrt{P_{11} P_{22} - \varepsilon} \]

end

end

**** Computational delay

The Kalman Filter provides and for the next iteration. If the computational delay, \( d \), is known, interpolation can be used:

\[ \hat{X}_{k+d} = d(\hat{X}_{k+d} - \hat{X}_k) + \hat{X}_k \]
where $0 < d \leq 1$

However, most just use $\hat{X}_{k+1}$ to output to the system that uses the state estimate.

**** Incorrect $R_k$

If the measurement noise matrix, $R_k$, is incorrect, suboptimal estimates will be generated by the Kalman Filter.

**** Incorrect $Q_k$

If the process noise matrix, $Q_k$, is incorrect, the Kalman Filter State Estimates will be suboptimal, an incorrect $Q_k$ will, in general, result in a larger negative impact on the Kalman Filter results than an incorrect $R_k$. 