Polynomial Kalman Filters
Polynomial Kalman Filters
Overview

• Kalman filtering equations
  - Scalar derivation
• Polynomial Kalman filter without process noise
• Comparing recursive least squares filter to Kalman filter
• Properties of polynomial Kalman filters
• Initial covariance matrix
• Polynomial Kalman filter with process noise
• Example of tracking a falling object
• A Kalman filter for accelerometer testing problem
Filtering Equations
General Continuous Equations

Model of real world must be placed in state space form
\[
x' = Fx + Gu + w
\]
Often this is fudge factor

Where \( u \) is known and we are interested in estimating \( x \)

Process noise matrix related to process noise
\[
Q = E[ww^T]
\]
Matrix of spectral densities

Measurements must be linearly related to states
\[
z = Hx + v
\]

Measurement noise matrix related to measurement noise
\[
R = E[vv^T]
\]
Matrix of spectral densities
Modifying Real World Equations so Discrete Kalman Filter Can be Built

Fundamental matrix

\[ \Phi(t) = e^{\int_0^t (sI - F) \, dt} \]

Laplace transform method

\[ \Phi(t) = e^{Ft} = I + Ft + \frac{(Ft)^2}{2!} + \cdots + \frac{(Ft)^n}{n!} + \cdots \]

Taylor series expansion

Discrete transition or fundamental matrix

\[ \Phi_k = \Phi(T_s) \]

Discrete measurement equation

\[ z_k = Hx_k + v_k \]

Discrete measurement noise matrix

\[ R_k = E(v_kv_k^T) \]

Matrix of variances
Discrete Kalman Filter

Filter formula

\[ \hat{x}_k = \Phi_k \hat{x}_{k-1} + G_k u_{k-1} + K_k (z_k - H \Phi_k \hat{x}_{k-1} - H G_k u_{k-1}) \]

where

\[ G_k = \int_0^{T_s} \Phi(\tau) G d\tau \]

Filter gains are obtained from Riccati equations

\[ M_k = \Phi_k P_{k-1} \Phi_k^T + Q_k \]

\[ K_k = M_k H^T (HM_k H^T + R_k)^{-1} \]

\[ P_k = (I - K_k H) M_k \]

where

\[ Q_k = \int_0^{T_s} \Phi(\tau) Q \Phi(\tau)^T d\tau \]

Don’t estimate this

We supply initial covariance matrix to get started
Derivation of **Scalar** Riccati Equations - 1

**Scalar form of Kalman filter if there is no known disturbance**

\[ \hat{x}_k = \Phi_k \hat{x}_{k-1} + K_k (z_k - H\Phi_k \hat{x}_{k-1}) \]

**Scalar measurement equation**

\[ z_k = Hx_k + v_k \]

**Error in the estimate**

\[ \tilde{x}_k = x_k - \hat{x}_k = x_k - \Phi_k \hat{x}_{k-1} - K_k (z_k - H\Phi_k \hat{x}_{k-1}) \]

**Express measurement in terms of state**

\[ \tilde{x}_k = x_k - \Phi_k \hat{x}_{k-1} - K_k (Hx_k + v_k - H\Phi_k \hat{x}_{k-1}) \]

**Since our model of the real world is given by**

\[ x_k = \Phi_k x_{k-1} + w_k \]

**We can say**

\[ \tilde{x}_k = \Phi_k x_{k-1} + w_k - \Phi_k \hat{x}_{k-1} - K_k (H\Phi_k x_{k-1} + Hw_k + v_k - H\Phi_k \hat{x}_{k-1}) \]
Derivation of Scalar Riccati Equations - 2

From previous slide

\[ \tilde{x}_k = \Phi_k \hat{x}_{k-1} + w_k - \Phi_k \tilde{x}_{k-1} - K_k (H \Phi_k \hat{x}_{k-1} + Hw_k + v_k - H \Phi_k \tilde{x}_{k-1}) \]

Since by definition

\[ \tilde{x}_k = x_k - \hat{x}_k \]

We can say that

\[ \tilde{x}_{k-1} = x_{k-1} - \hat{x}_{k-1} \]

Substitution and combining similar terms yields

\[ \tilde{x}_k = (1 - K_k H) \tilde{x}_{k-1} \Phi_k + (1 - K_k H) w_k - K_k v_k \]

Squaring both sides and taking expectations yields

\[ P_k = (1 - K_k H)^2 (P_{k-1} \Phi_k^2 + Q_k) + K_k^2 R_k \]

Where

\[ P_k = E(\tilde{x}_k^2) \quad \text{New definition} \]

\[ Q_k = E(w_k^2) \]

\[ R_k = E(v_k^2) \]

\[ E(\tilde{x}_{k-1} w_k) = 0 \]

\[ E(\tilde{x}_{k-1} v_k) = 0 \]

\[ E(w_k v_k) = 0 \]
Derivation of **Scalar Riccati Equations - 3**

From previous slide

\[ P_k = (1 - K_k H)^2( P_{k-1} \Phi_k^2 + Q_k) + K_k^2 R_k \]

By defining

\[ M_k = P_{k-1} \Phi_k^2 + Q_k \]

This is analogous to first Riccati equation

And substituting we get

\[ P_k = (1 - K_k H)^2 M_k + K_k^2 R_k \]

To find gain that will minimize the variance of the error in the estimate we can use calculus (i.e., take derivative and set to zero)

\[ \frac{\partial P_k}{\partial K_k} = 0 = 2(1 - K_k H)M_k(-H) + 2K_k R_k \]

Solving for the gain yields

\[ K_k = \frac{M_k H}{H^2 M_k + R_k} = M_k H( H^2 M_k + R_k)^{-1} \]

This is analogous to second Riccati equation
Recall from previous slide

\[ P_k = (1 - K_k H)^2 M_k + K_k^2 R_k \]

\[ K_k = \frac{M_k H}{H^2 M_k + R_k} = M_k H (H^2 M_k + R_k)^{-1} \]  \hspace{1cm} \text{Optimal gain equation}

Substitute optimal gain into variance equation

\[ P_k = (1 - M_k H^2)^2 M_k + \left( \frac{M_k H}{H^2 M_k + R_k} \right)^2 R_k \]

Which simplifies to

\[ P_k = \frac{R_k M_k}{H^2 M_k + R_k} = \frac{R_k K_k}{H} \]  \hspace{1cm} \text{Variance equation}

By inverting optimal gain equation

\[ K_k R_k = M_k H - H^2 M_k K_k \]

Substitute back into variance equation

\[ P_k = \frac{R_k K_k}{H} = \frac{M_k H - H^2 M_k K_k}{H} = M_k - HM_k K_k \]

or

\[ P_k = (1 - K_k H) M_k \]  \hspace{1cm} \text{This is analogous to third Riccati equation}
Riccati Equation Summary

Matrix Riccati equations

\[ M_k = \Phi_k P_{k-1} \Phi_k^T + Q_k \]
\[ K_k = M_k H^T (H M_k H^T + R_k)^{-1} \]
\[ P_k = (I - K_k H) M_k \]

Scalar Riccati equations

\[ M_k = P_{k-1} \Phi_k^2 + Q_k \]
\[ K_k = M_k H (H^2 M_k + R_k)^{-1} \]
\[ P_k = (1 - K_k H) M_k \]
Polynomial Kalman Filter Without Process Noise
State Equations For Different Order Polynomials

Constant signal
\[ x = a_0 \quad \Rightarrow \quad \dot{x} = 0 \quad \Rightarrow \quad \dot{x} = Fx \quad \Rightarrow \quad F = 0 \]

Ramp signal
\[ x = a_0 + a_1 t \quad \Rightarrow \quad \dot{x} = a_1 \quad \Rightarrow \quad \begin{bmatrix} \dot{x} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad \Rightarrow \quad F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \]

Parabolic Signal
\[ x = a_0 + a_1 t + a_2 t^2 \quad \Rightarrow \quad \dot{x} = a_1 + 2a_2 t \quad \Rightarrow \quad \begin{bmatrix} \ddots \\ \ddots \\ \ddots \\ \ddots \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddots \end{bmatrix} \quad \Rightarrow \quad F = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \]
We are Assuming Measurement is Polynomial Signal Plus Noise

**Constant signal**

\[ z = x^* = a_0 + \text{noise} \quad \sigma_{\text{noise}} = \sigma_n \]
\[ z = x + n \quad \rightarrow \quad H = 1 \]

**Ramp signal**

\[ z = x^* = a_0 + a_1 t + \text{noise} \quad \sigma_{\text{noise}} = \sigma_n \]
\[ z = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + n \quad \rightarrow \quad H = \begin{bmatrix} 1 & 0 \end{bmatrix} \]

**Parabolic signal**

\[ z = x^* = a_0 + a_1 t + a_2 t^2 + \text{noise} \quad \sigma_{\text{noise}} = \sigma_n \]
\[ z = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + n \quad \rightarrow \quad H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \]
Example of Deriving a Fundamental Matrix

For first-order system
\[ F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \]

Squaring the systems dynamics matrix
\[ F^2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \]

Fundamental matrix can be found by Taylor series expansion
\[ \Phi(t) = e^{Ft} = I + Ft + \frac{(Ft)^2}{2!} + \ldots + \frac{(Ft)^n}{n!} + \ldots \]

Only two terms are required because
\[ \Phi(t) = e^{Ft} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} t + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \frac{t^2}{2} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \]

Therefore discrete fundamental matrix is
\[ \Phi_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \]
## Important Matrices for Different Order Polynomial Kalman Filters

<table>
<thead>
<tr>
<th>Order</th>
<th>Systems Dynamics</th>
<th>Fundamental</th>
<th>Measurement</th>
<th>Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$F = 1$</td>
<td>$\Phi_k = 1$</td>
<td>$H = 1$</td>
<td>$R_k = \sigma_n^2$</td>
</tr>
<tr>
<td>1</td>
<td>$F = \begin{bmatrix} 0 &amp; 1 \ 0 &amp; 0 \end{bmatrix}$</td>
<td>$\Phi_k = \begin{bmatrix} 1 &amp; T_s \ 0 &amp; 1 \end{bmatrix}$</td>
<td>$H = [1 \ 0]$</td>
<td>$R_k = \sigma_n^2$</td>
</tr>
<tr>
<td>2</td>
<td>$F = \begin{bmatrix} 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$\Phi_k = \begin{bmatrix} 1 &amp; T_s &amp; .5T_s^2 \ 0 &amp; 1 &amp; T_s \ 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$H = [1 \ 0 \ 0]$</td>
<td>$R_k = \sigma_n^2$</td>
</tr>
</tbody>
</table>
Comparing Zeroth-Order Recursive Least Squares and Kalman Filters
Zeroth-Order Polynomial Kalman Filter

Kalman filter equation

\[ \hat{x}_k = \Phi_k \hat{x}_{k-1} + K_k (z_k - H \Phi_k \hat{x}_{k-1}) \]

Substituting matrices for zeroth-order filter

\[ \hat{x}_k = \hat{x}_{k-1} + K_{1_k} (x^*_k - \hat{x}_{k-1}) \]

If we define the residual to be

\[ \text{RES}_k = x^*_k - \hat{x}_{k-1} \]

Then zeroth-order Kalman filter becomes

\[ \hat{x}_k = \hat{x}_{k-1} + K_{1_k} \text{RES}_k \]

*This is identical to equation for zeroth-order recursive least squares filter!*
Solving for Gain of Zeroth-Order Polynomial Kalman Filter - 1

For zero process noise Ricatti equations are

\[ M_k = \Phi_k P_{k-1} \Phi_k^T + Q_k = \Phi_k P_{k-1} \Phi_k^T \]

\[ K_k = M_k H^T (H M_k H^T + R_k)^{-1} \]

\[ P_k = (I - K_k H) M_k \]

Assume initial covariance is infinite

\[ P_0 = \infty \]

From first Riccati equation

\[ M_1 = \Phi_1 P_0 \Phi_1^T = 1 \rightarrow 1 = \infty \]

From second Riccati equation

\[ K_1 = M_1 H^T (H M_1 H^T + R_1)^{-1} = \frac{M_1}{M_1 + \sigma_n^2} = \frac{\infty}{\infty + \sigma_n^2} = 1 \]

Same gain as recursive least squares filter with \( k=1 \)

From third Riccati equation

\[ P_1 = (I - K_1 H) M_1 = \left[ 1 - \frac{M_1}{M_1 + \sigma_n^2} \right] M_1 = \frac{\sigma_n^2 M_1}{M_1 + \sigma_n^2} = \sigma_n^2 \]

Same variance as recursive least squares filter with \( k=1 \)

Fundamentals of Kalman Filtering: A Practical Approach
Solving for Gain of Zeroth-Order Polynomial
Kalman Filter - 2

Second iteration of Riccati equation

\[ M_2 = \Phi_2 P_1 \Phi_2^T = 1 \sigma_n^2 + 1 = \sigma_n^2 \]

\[ K_2 = \frac{M_2}{M_2 + \sigma_n^2} = \frac{\sigma_n^2}{\sigma_n^2 + \sigma_n^2} = 0.5 \]

\[ P_2 = \frac{\sigma_n^2 M_2}{M_2 + \sigma_n^2} = \frac{\sigma_n^2 \sigma_n^2}{\sigma_n^2 + \sigma_n^2} = \frac{\sigma_n^2}{2} \]

Same gain as recursive least squares filter with \( k=2 \)

Same variance as recursive least squares filter with \( k=2 \)

Third iteration of Riccati equation

\[ M_3 = P_2 = \frac{\sigma_n^2}{2} \]

\[ K_3 = \frac{M_3}{M_3 + \sigma_n^2} = \frac{0.5 \sigma_n^2}{0.5 \sigma_n^2 + \sigma_n^2} = \frac{1}{3} \]

\[ P_3 = \frac{\sigma_n^2 M_3}{M_3 + \sigma_n^2} = \frac{\sigma_n^2 \cdot 0.5 \sigma_n^2}{0.5 \sigma_n^2 + \sigma_n^2} = \frac{\sigma_n^2}{3} \]

Same gain and variance as recursive Least squares filter with \( k=3 \)

Fourth iteration of Riccati equation

\[ M_4 = P_3 = \frac{\sigma_n^2}{3} \]

\[ K_4 = \frac{M_4}{M_4 + \sigma_n^2} = \frac{0.333 \sigma_n^2}{0.333 \sigma_n^2 + \sigma_n^2} = \frac{1}{4} \]

\[ P_4 = \frac{\sigma_n^2 M_4}{M_4 + \sigma_n^2} = \frac{\sigma_n^2 \cdot 0.333 \sigma_n^2}{0.333 \sigma_n^2 + \sigma_n^2} = \frac{\sigma_n^2}{4} \]

Same gain and variance as recursive Least squares filter with \( k=4 \)
Summary of Comparison

Thus we can see that when the zeroth-order polynomial Kalman filter has zero process noise and infinite initial covariance matrix, it had the same gains and variance predictions as the zeroth-order recursive least squares filter.
Comparing First-Order Recursive Least Squares and Kalman Filters
First-Order Polynomial Kalman Filter

Kalman filtering equation

\[ \hat{x}_k = \Phi_k \hat{x}_{k-1} + K_k (z_k - H \Phi_k \hat{x}_{k-1}) \]

Substitution of matrices yields

\[
\begin{bmatrix}
\hat{x}_k \\
\hat{x}_k
\end{bmatrix} =
\begin{bmatrix}
1 & T_s \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
\hat{x}_{k-1} \\
\hat{x}_{k-1}
\end{bmatrix} +
\begin{bmatrix}
K_{1s} \\
K_{2s}
\end{bmatrix}
\begin{bmatrix}
x_k^* - 1 \\
1 & 0
\end{bmatrix}
\begin{bmatrix}
\hat{x}_{k-1} \\
\hat{x}_{k-1}
\end{bmatrix}
\]

Multiplying out the matrices

\[ \hat{x}_k = \hat{x}_{k-1} + T_s \hat{x}_{k-1} + K_{1s} (x_k^* - \hat{x}_{k-1} - T_s \hat{x}_{k-1}) \]

\[ \hat{x}_k = \hat{x}_{k-1} + K_{2s} (x_k^* - \hat{x}_{k-1} - T_s \hat{x}_{k-1}) \]

If we define the residual to be

\[ \text{RES}_k = x_k^* - \hat{x}_{k-1} - T_s \hat{x}_{k-1} \]

The filter becomes

\[ \hat{x}_k = \hat{x}_{k-1} + T_s \hat{x}_{k-1} + K_{1s} \text{RES}_k \]

\[ \hat{x}_k = \hat{x}_{k-1} + K_{2s} \text{RES}_k \]

Identical to equations of first-order recursive least squares filter
Review of Equations for Gains and Variances of First-Order Recursive Least Squares Filter

**Gains**

\[ K_{1k} = \frac{2(2k-1)}{k(k+1)} \quad k=1, 2, \ldots, n \]

\[ K_{2k} = \frac{6}{k(k+1)}T_s \]

**Variances of errors in the estimates**

\[ P_{11k} = \frac{2(2k-1)\sigma_n^2}{k(k+1)} \]

\[ P_{22k} = \frac{12\sigma_n^2}{k(k^2-1)}T_s^2 \]
MATLAB Simulation to Compare Both Filters - 1

ORDER=2;  
TS=1.;  
SIGNOISE=1.;  
PHI=[1 TS;0 1];  
P=[99999999 0;0 99999999];  
IDNP=eye(ORDER);  
H=[1 0];  
HT=H';  
R=SIGNOISE^2;  
PHIT=PHI';  
count=0;  
for XN=1:100

PHIP=PHI*P;  
M=PHIP*PHIT;  
MHT=M*HT;  
HMHT=H*MHT;  
HMHTR=HMHT+R;  
HMHTRINV=inv(HMHTR);  
K=MHT*HMHTRINV;  
KH=K*H;  
IKH=IDNP-KH;  
P=IKH*M;  
if XN<2

P11GM=9999999999.;  
P22GM=9999999999.;  
else

P11GM=2.*(2.*XN-1).*SIGNOISE*SIGNOISE/(XN*(XN+1.));  
P22GM=12.*SIGNOISE*SIGNOISE/(XN*(XN*XN-1.)*TS*TS);  
end

SP11=sqrt(P(1,1));  
SP22=sqrt(P(2,2));  
SP11GM=sqrt(P11GM);  
SP22GM=sqrt(P22GM);  
K1GM=2.*(2.*XN-1.)/(XN*(XN+1.));  
K2GM=6./(XN*(XN+1.)*TS);  
K1=K(1,1);  
K2=K(2,1);
MATLAB Simulation to Compare Both Filters - 2

count=count+1;
ArrayXN(count)=XN;
ArrayK1(count)=K1;
ArrayK1GM(count)=K1GM;
ArrayK2(count)=K2;
ArrayK2GM(count)=K2GM;
ArraySP11(count)=SP11;
ArraySP11GM(count)=SP11GM;
ArraySP22(count)=SP22;
ArraySP22GM(count)=SP22GM;

disp 'Simulation finished'
end
figure
plot(ArrayXN,ArraySP11,ArrayXN,ArraySP11GM),grid
xlabel('Number of Measurements')
ylabel('Error in Estimate of First State')
axis([0 100 0 1])
figure
plot(ArrayXN,ArraySP22,ArrayXN,ArraySP22GM),grid
xlabel('Number of Measurements')
ylabel('Error in Estimate of Second State')
axis([0 100 0 .1])
figure
plot(ArrayXN,ArrayK1,ArrayXN,ArrayK1GM),grid
xlabel('Number of Measurements')
ylabel('First Kalman Gain')
axis([0 100 0 1])
figure
plot(ArrayXN,ArrayK2,ArrayXN,ArrayK2GM),grid
xlabel('Number of Measurements')
ylabel('Second Kalman Gain')
axis([0 100 0 .1])
clc
output=[ArrayXN',ArrayK1',ArrayK1GM',ArrayK2',ArrayK2GM'];
save datfil output -ascii
output=[ArrayXN',ArraySP11',ArraySP11GM',ArraySP22',ArraySP22GM'];
save covfil output -ascii
disp 'Simulation finished'

Saving information so it can be plotted and displayed

Generating plots

Write data to files
First-Order Polynomial Kalman and Recursive Least Squares Filters Have Identical Standard Deviations for Errors in the Estimate of the First State
First-Order Polynomial Kalman and Recursive Least Squares Filters Have Identical Standard Deviations for Errors in the Estimate of the Second State.
First Gain of First-Order Polynomial Kalman and Recursive Least Squares Filters are Identical
Second Gain of First-Order Polynomial Kalman and Recursive Least SquaresFilters are Identical

![Graph showing the comparison between Second Gain of First-Order Polynomial Kalman and Recursive Least Squares Filters with respect to the number of measurements. The graph indicates that the gains are identical for certain conditions.](image)

First-Order
\[ T_s = 1 \text{ S} \]

Recursive Least Squares and Kalman (Q=0, P_0=∞)
Summary of Comparison

Thus we can see that when the first-order polynomial Kalman filter has zero process noise and infinite initial covariance matrix, it had the same gains and variance predictions as the first-order recursive least squares filter.
Comparing Second-Order Recursive Least Squares and Kalman Filters
Second-Order Polynomial Kalman Filter

Kalman filtering equation

$$\hat{x}_k = \Phi_k \hat{x}_{k-1} + K_k(z_k - H \Phi_k \hat{x}_{k-1})$$

Substitution of matrices yields

$$\begin{bmatrix} \hat{x}_k \\ \hat{\hat{x}}_k \end{bmatrix} = \begin{bmatrix} 1 & T_s & .5 T_s^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\hat{x}}_{k-1} \end{bmatrix} + \begin{bmatrix} K_{1_k} \\ K_{2_k} \\ K_{3_k} \end{bmatrix} \begin{bmatrix} x^k - [1 0 0] \\ 0 1 T_s \\ 0 0 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\hat{x}}_{k-1} \end{bmatrix}$$

Multiplying out the matrices

$$\hat{x}_k = \hat{x}_{k-1} + T_s \hat{x}_{k-1} + .5 T_s^2 \hat{x}_{k-1} + K_{1_k}(x^k - \hat{x}_{k-1} - T_s \hat{x}_{k-1} - .5 T_s^2 \hat{x}_{k-1})$$

$$\hat{x}_k = \hat{x}_{k-1} + T_s \hat{x}_{k-1} + K_{2_k}(x^k - \hat{x}_{k-1} - T_s \hat{x}_{k-1} - T_s \hat{x}_{k-1})$$

$$\hat{x}_k = \hat{x}_{k-1} + K_{3_k}(x^k - \hat{x}_{k-1} - T_s \hat{x}_{k-1} - T_s \hat{x}_{k-1})$$

If we define the residual to be

$$\text{RES}_k = x^k - \hat{x}_{k-1} - T_s \hat{x}_{k-1}$$

The filter becomes

$$\hat{x}_k = \hat{x}_{k-1} + T_s \hat{x}_{k-1} + .5 T_s^2 \hat{x}_{k-1} + K_{1_k}\text{RES}_k$$

$$\hat{x}_k = \hat{x}_{k-1} + T_s \hat{x}_{k-1} + K_{2_k}\text{RES}_k$$

$$\hat{x}_k = \hat{x}_{k-1} + K_{3_k}\text{RES}_k$$

Identical to equations of second-order recursive least squares filter
Review of Equations for Gains and Variances of Second-Order Recursive Least Squares Filter

Gains

\[ K_{1k} = \frac{3(3k^2 - 3k + 2)}{k(k+1)(k+2)} \]  \quad k=1,2,...,n

\[ K_{2k} = \frac{18(2k-1)}{k(k+1)(k+2)T_s} \]

\[ K_{3k} = \frac{60}{k(k+1)(k+2)T_s^2} \]

Variances of errors in the estimates

\[ P_{11k} = \frac{3(3k^2 - 3k + 2)\sigma_n^2}{k(k+1)(k+2)} \]

\[ P_{22k} = \frac{12(16k^2 - 30k + 11)\sigma_n^2}{k(k^2 - 1)(k^2 - 4)T_s^2} \]

\[ P_{33k} = \frac{720\sigma_n^4}{k(k^2 - 1)(k^2 - 4)T_s^4} \]
MATLAB Simulation to Compare Both Filters - 1

```
ORDER=3;
TS=1.;
SIGNOISE=1.;
PHI=[1 TS .5*TS*TS;0 1 TS;0 0 1];
P=[99999999 0 0; 999999999 0 0; 0 0 999999999];
IDNP=eye(ORDER);
H=[1 0 0];
HT=H';
R=SIGNOISE^2;
PHIT=PHI';
count=0;
for XN=1:100

PHIP=PHI*P;
M=PHIP*PHIT;
MHT=M*HT;
HMHT=H*MHT;
HMHTR=HMHT+R;
HMHTRINV=inv(HMHTR);
K=MHT*HMHTRINV;
KH=K*H;
IKH=IDNP-KH;
P=IKH*M;
if XN<3
    P11GM=9999999999.;
    P22GM=9999999999.;
    P33GM=9999999999.;
else
    P11GM=(3*(3*XN*XN-3*XN+2)/(XN*(XN+1)*(XN+2))*SIGNOISE/2;
    P22GM=(12*(16*XN*XN-30*XN+11)/(XN*(XN*XN-1)*(XN*XN-4)*TS*TS)*SIGNOISE/2;
    P33GM=(720/(XN*(XN*XN-1)*(XN*XN-4)*TS*TS*TS)*SIGNOISE/2;
end
```

Fundamental & initial covariance matrices for second-order polynomial Kalman filter

Riccati equations

Least squares filter covariances
MATLAB Simulation to Compare Both Filters - 2

\[ SP11 = \text{sqrt}(P(1,1)); \]
\[ SP22 = \text{sqrt}(P(2,2)); \]
\[ SP33 = \text{sqrt}(P(3,3)); \]
\[ SP11GM = \text{sqrt}(P_{11GM}); \]
\[ SP22GM = \text{sqrt}(P_{22GM}); \]
\[ SP33GM = \text{sqrt}(P_{33GM}); \]
\[ K1GM = 3 \times (3 \times XN \times XN - 3 \times XN + 2) / (XN \times (XN + 1) \times (XN + 2)); \]
\[ K2GM = 18 \times (2 \times XN - 1) / (XN \times (XN + 1) \times (XN + 2) \times TS); \]
\[ K3GM = 60 / (XN \times (XN + 1) \times (XN + 2) \times TS \times TS); \]
\[ K1 = K(1,1); \]
\[ K2 = K(2,1); \]
\[ K3 = K(3,1); \]
\[ \text{if } XN \geq 3 \]
\[ \text{count} = \text{count} + 1; \]
\[ \text{ArrayXN(count)} = XN; \]
\[ \text{ArrayK1(count)} = K1; \]
\[ \text{ArrayK1GM(count)} = K1GM; \]
\[ \text{ArrayK2(count)} = K2; \]
\[ \text{ArrayK2GM(count)} = K2GM; \]
\[ \text{ArrayK3(count)} = K3; \]
\[ \text{ArrayK3GM(count)} = K3GM; \]
\[ \text{ArraySP11(count)} = SP11; \]
\[ \text{ArraySP11GM(count)} = SP11GM; \]
\[ \text{ArraySP22(count)} = SP22; \]
\[ \text{ArraySP22GM(count)} = SP22GM; \]
\[ \text{ArraySP33(count)} = SP33; \]
\[ \text{ArraySP33GM(count)} = SP33GM; \]
\[ \text{end} \]

Least squares filter gains

Saving information so it can be plotted and displayed
Covariance Matrix Projections of First State are Identical for Both Second-Order Polynomial Kalman and Recursive Least Squares Filters
Covariance Matrix Projections of Second State are Identical for Both Second-Order Polynomial Kalman and Recursive Least Squares Filters
Covariance Matrix Projections of Third State are Identical for Both Second-Order Polynomial Kalman and Recursive Least Squares Filters

![Graph showing error in estimate of third state for both filters as a function of number of measurements. The graph depicts a sharp decrease in error with increasing number of measurements for both filters, indicating the effectiveness of both methods.](image-url)
First Gain of Second-Order Polynomial Kalman and Recursive Least Squares Filters are Identical

Recursive Least Squares and Kalman (Q=0, P_0=\infty)
Second Gain of Second-Order Polynomial Kalman and Recursive Least Squares Filters are Identical
Third Gain of Second-Order Polynomial Kalman and Recursive Least Squares Filters are Identical

Recursive Least Squares and Kalman ($Q=0$, $P_0=\infty$)
Summary of Comparison

Thus we can see that when the second-order polynomial Kalman filter has zero process noise and infinite initial covariance matrix, it had the same gains and variance predictions as the second-order recursive least squares filter.
Comparing Different Order Polynomial Kalman Filters
Errors in Estimate of First State are Smaller With Lower Order Filters

![Graph showing errors in estimate of first state with different order filters]

- Zeroth-Order
- First-Order
- Second-Order

Key:
- $T_s = 1 \text{ s}$
- $\sigma_N = 1$
- $Q = 0$
Errors in Estimate of Second State are Smaller With Lower Order Filters

- First-Order
- Second-Order
Errors in Estimate of Third State Decrease as Number of Measurements Taken Increase

![Graph showing the error in estimate of third state decreases as the number of measurements increases. The graph has a logarithmic scale on the x-axis for the number of measurements and a linear scale on the y-axis for the error in estimate. The graph includes a legend with the following parameters:

- $T_s = 1$ S
- $\sigma_n = 1$
- $Q = 0$]
Initial Covariance Matrix
Errors in Estimates of First State of a Zeroth-Order Polynomial Kalman Filter are Fairly Insensitive to the Initial Covariance Matrix

- $P_0 = \infty$ and $P_0 = 100$
- $P_0 = 0.1$
- $P_0 = 0$
- $T_s = 1$, $\sigma = 1$

Number of Measurements

Error in Estimate of First State, $P_1$
Errors in Estimates of First State of a First-Order Polynomial Kalman Filter are Also Fairly Insensitive to the Initial Covariance Matrix
Errors in Estimates of Second State of a First-Order Polynomial Kalman Filter are Fairly Insensitive to the Initial Covariance Matrix

![Graph showing errors in estimates of second state](image)

- $P_0 = \infty$ and $P_0 = 100$
- $P_0 = 1$
- $P_0 = 0$
- $P_0 = 0.1$

First-Order $T_s = 1$ S, $\sigma_N = 1$
Errors in Estimates of First State of a Second-Order Polynomial Kalman Filter are Also Fairly Insensitive to the Initial Covariance Matrix
Errors in Estimates of Second State of a Second-Order Polynomial Kalman Filter are Also Fairly Insensitive to the Initial Covariance Matrix

\[ P_0 = \infty \text{ and } P_0 = 100 \]

\[ P_0 = 1 \]

\[ P_0 = 0 \]

Number of Measurements

Error in Estimate of Second State, \( P_{2|k} \)

Second-Order
\[ T_s = 1 \text{ S}, \sigma_N = 1 \]
Errors in Estimates of Third State of a Second-Order Polynomial Kalman Filter are Fairly Insensitive to the Initial Covariance Matrix

- $P_0 = \infty$ and $T_s = 1$, $\sigma_N = 1$
- $P_0 = 100$
- $P_0 = 1$
- $P_0 = 0$
We can conclude that the theoretical performance (i.e., standard deviation of the error in the estimate) of all the polynomial Kalman filters are insensitive to the initial value of the covariance matrix.
Polynomial Kalman Filter With Process Noise
Modifying State Equations to Include Process Noise

Riccati equations with process noise

\[ M_k = \Phi_k P_{k-1} \Phi_k^T + Q_k \]

\[ K_k = M_k H^T (HM_k H^T + R_k)^{-1} \]

\[ P_k = (I - K_k H) M_k \]

**Constant signal**

\[ \dot{x} = u_s \quad \rightarrow \quad Q = E(u_s u_s^T) = \Phi_s \]

**Ramp Signal**

\[
\begin{bmatrix}
\dot{x} \\
\ddot{x}
\end{bmatrix}
= 
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
\dot{x}
\end{bmatrix}
+ 
\begin{bmatrix}
0 \\
u_s
\end{bmatrix}
\quad \rightarrow \quad Q = E \begin{bmatrix}
0 & 0 \\
u_s & 0
\end{bmatrix}
= \begin{bmatrix}
0 & 0 \\
0 & \Phi_s
\end{bmatrix}
\]

**Parabolic Signal**

\[
\begin{bmatrix}
\dot{x} \\
\ddot{x} \\
\dddot{x}
\end{bmatrix}
= 
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
\dot{x} \\
\ddot{x}
\end{bmatrix}
+ 
\begin{bmatrix}
0 \\
0 \\
u_s
\end{bmatrix}
\quad \rightarrow \quad Q = E \begin{bmatrix}
0 \\
0 \\
u_s
\end{bmatrix}
= \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & \Phi_s
\end{bmatrix}
\]

**Deriving discrete process noise matrix**

\[ Q_k = \int_0^{T_s} \Phi(\tau) Q \Phi^T(\tau) d\tau \]
The Discrete Process Noise Matrix Varies With System Order

<table>
<thead>
<tr>
<th>Order</th>
<th>Continuous Q</th>
<th>Fundamental</th>
<th>Discrete Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$Q = \Phi_s$</td>
<td>$\Phi_k = 1$</td>
<td>$Q_k = \Phi_s$</td>
</tr>
<tr>
<td>1</td>
<td>$Q = \Phi_s \begin{bmatrix} 0 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
<td>$\Phi_k = \begin{bmatrix} 1 &amp; T_s \ 0 &amp; 1 \end{bmatrix}$</td>
<td>$Q_k = \Phi_s \begin{bmatrix} \frac{T_s^3}{3} &amp; \frac{T_s^2}{2} \ \frac{T_s^2}{2} &amp; T_s \end{bmatrix}$</td>
</tr>
<tr>
<td>2</td>
<td>$Q = \Phi_s \begin{bmatrix} 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$\Phi_k = \begin{bmatrix} 1 &amp; T_s &amp; .5T_s^2 \ 0 &amp; 1 &amp; T_s \ 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$Q_k = \Phi_s \begin{bmatrix} \frac{T_s^5}{20} &amp; \frac{T_s^4}{8} &amp; \frac{T_s^3}{6} \ \frac{T_s^4}{8} &amp; \frac{T_s^3}{3} &amp; \frac{T_s^2}{2} \ \frac{T_s^3}{6} &amp; \frac{T_s^2}{2} &amp; T_s \end{bmatrix}$</td>
</tr>
</tbody>
</table>
Alternative Method For Finding Discrete Process Noise Matrix

If our model of the real world is given by

\[
\dot{x} = Fx + Gu + w, \quad Q_c = E[ww^T]
\]

The discrete process noise matrix can also be found by solving

\[
\dot{Q}_k = FQ_k + Q_k F^T + Q_c, \quad Q_k(0) = 0
\]

As an example we already know that for

\[
\begin{bmatrix}
\dot{x} \\
\dot{\dot{x}} \\
\ddot{\dot{x}}
\end{bmatrix} = 
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
\dot{x} \\
\ddot{x}
\end{bmatrix} + 
\begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix} u_s, \quad Q_c = \Phi_s
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

The discrete process noise matrix is given by

\[
Q_k = \Phi_s
\begin{bmatrix}
t^5 & t^4 & t^3 \\
t^2 & t^1 & t^0 \\
8 & 6 & 2 \\
3 & 2 & 1 \\
6 & 2 & 0
\end{bmatrix}
\]
Computing Discrete Process Noise Matrix-1

```fortran
IMPLICIT REAL*8(A-H)
IMPLICIT REAL*8(O-Z)
REAL*8 F(3,3),Q(3,3),QK(3,3),FQK(3,3),FT(3,3),QKFT(3,3)
REAL*8 FQKQKFT(3,3),QKD(3,3),QKOLD(3,3)
INTEGER ORDER
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
ORDER=3
T=0.
S=0.
H=.001
TS=.1
TF=10.
PHIS=1.
XJ=1.
DO 14 I=1,ORDER
  DO 14 J=1,ORDER
    F(I,J)=0.
    Q(I,J)=0.
    QK(I,J)=0.
14 CONTINUE
F(1,2)=1.
F(2,3)=1.
Q(3,3)=PHIS
CALL MATTRN(F,ORDER,ORDER,FT)
WHILE(T<=TF)
  DO 20 I=1,ORDER
    DO 20 J=1,ORDER
      QKOLD(I,J)=QK(I,J)
20 CONTINUE
```

F, Q_c and Q_k(0)

Second-Order Runge-Kutta Integration

Fundamentals of Kalman Filtering: A Practical Approach
Computing Discrete Process Noise Matrix-2

CALL MATMUL(F,ORDER,ORDER,QK,ORDER,ORDER,FQK)
CALL MATMUL(QK,ORDER,ORDER,FT,ORDER,ORDER,QKFT)
CALL MATADD(FQK,ORDER,ORDER,QKFT,FQKQKFT)
CALL MATADD(FQKQKFT,ORDER,ORDER,Q,QKD)
DO 50 I=1,ORDER
DO 50 J=1,ORDER
   QK(I,J)=QK(I,J)+H*QKD(I,J)
50 CONTINUE
T=T+H
CALL MATMUL(F,ORDER,ORDER,QK,ORDER,ORDER,FQK)
CALL MATMUL(QK,ORDER,ORDER,FT,ORDER,ORDER,QKFT)
CALL MATADD(FQK,ORDER,ORDER,QKFT,FQKQKFT)
CALL MATADD(FQKQKFT,ORDER,ORDER,Q,QKD)
DO 60 I=1,ORDER
DO 60 J=1,ORDER
   QK(I,J)=.5*(QKOLD(I,J)+QK(I,J)+H*QKD(I,J))
60 CONTINUE
S=S+H
IF(S>=TS-.00001)THEN
   S=0.
   WRITE(9,*),T,QK(1,1),QK(1,2),QK(1,3),QK(2,1),
   1 QK(2,2),QK(2,3),QK(3,1),QK(3,2),
   2 QK(3,3)
   WRITE(1,*),T,QK(1,1),QK(1,2),QK(1,3),QK(2,1),
   1 QK(2,2),QK(2,3),QK(3,1),QK(3,2),
   2 QK(3,3)
ENDIF
END DO
PAUSE
CLOSE(1)
END
Elements of Computer Generated Discrete Process Noise Matrix Agree With Theory

- $Q_{k}(1,1)$ with $t^{5/20}$
- $Q_{k}(1,2)$ with $t^{4/8}$
- $Q_{k}(1,3)$ with $t^{3/6}$
- $Q_{k}(2,1)$ with $t^{4/8}$
- $Q_{k}(2,2)$ with $t^{3/3}$
- $Q_{k}(2,3)$ with $t^{2/2}$
- $Q_{k}(3,1)$ with $t^{3/6}$
- $Q_{k}(3,2)$ with $t^{2/2}$
- $Q_{k}(3,3)$ with $t$
Error in Estimate of the State Degrades as Process Noise Decreases for Zeroth-Order Filter

The graph illustrates the error in the estimate of the first state, $\Phi_s$, as a function of the number of measurements for different values of $\Phi_s$: $\Phi_s = 0$, $\Phi_s = 1$, and $\Phi_s = 100$. The error decreases as the number of measurements increases, indicating that the estimate improves with more data when process noise decreases. The graph is labeled with 'Zeroth-Order $T_s = 1$ S, $\sigma_N = 1$'.
Increasing the Process Noise Increases the Error in the Estimate of the First State of a First-Order Polynomial Kalman Filter
Increasing the Process Noise Increases the Error in the Estimate of the Second State of a First-Order Polynomial Kalman Filter

![Graph showing the relationship between process noise and error in estimate]

**Graph Details:**
- **First-Order Polynomial Kalman Filter**
- **Parameters:**
  - \( T = 1 \) s, \( \sigma_N = 1 \)
- **Equations:**
  - \( \Phi_s = 0 \)
  - \( \Phi_s = 1 \)
  - \( \Phi_s = 10 \)
  - \( \Phi_s = 100 \)
Increasing the Process Noise Increases the Error in the Estimate of the First State of a Second-Order Polynomial Kalman Filter
Increasing the Process Noise Increases the Error in the Estimate of the Second State of a Second-Order Polynomial Kalman Filter

Number of Measurements

Error in Estimate of Second State, $P_{2^2}$

$T_s = 1$, $\sigma_N = 1$

Second-Order $\Phi_s = 0$, $\Phi_s = 10$, $\Phi_s = 100$
Increasing the Process Noise Increases the Error in the Estimate of the Third State of a Second-Order Polynomial Kalman Filter

![Graph showing the effect of process noise on the estimated error of the third state. The graph plots error versus number of measurements for different values of process noise (Φ_s = 0, 10, 100).]
Example of Kalman Filter Tracking a Falling Object
Radar Tracking Falling Object

From basic physics

\[ x = 400000 - 6000t - \frac{gt^2}{2} \]

Velocity of object can be found by differentiating

\[ x = -6000 - gt \]

Second-order process

Radar measures altitude with standard deviation of 1000 ft

Desire to track object and estimate altitude and velocity
MATLAB Simulation of Second-Order Polynomial Kalman Filter and Falling Object-1

ORDER =3;
PHIS=0.;
TS=.1;
A0=400000;
A1=-6000.;
A2=-16.1;
XH=0;
XDH=0;
XDDH=0;
SIGNOISE=1000.;
PHI=[1 TS .5*TS^2;0 1 TS;0 0 1];
P=[99999999 0 0;0 999999999 0;0 0 999999999];
IDNP=eye(ORDER);
Q=zeros(ORDER);
H=[1 0 0];
HT=H';
R=SIGNOISE^2;
PHIT=PHI';
Q(1,1)=PHIS*TS^5/20;
Q(1,2)=PHIS*TS^4/8;
Q(1,3)=PHIS*TS^3/6;
Q(2,1)=Q(1,2);
Q(2,2)=PHIS*TS^3/3;
Q(2,3)=PHIS*TS*TS/2;
Q(3,1)=Q(1,3);
Q(3,2)=Q(2,3);
Q(3,3)=PHIS*TS;
count=0;
for T=0:TS:30
    PHIP=PHI*P;
    PHIPPHIT=PHIP*PHIT;
    M=PHIPPHIT+Q;
    MHT=M*HT;
    HMHT=H*MHT;
    HMHTR=HMHT+R;
    HMHTRINV=inv(HMHTR);
    K=MHT*HMHTRINV;
    KH=K*H;
    IKH=IDNP-KH;
    P=IKH*M;
end
MATLAB Simulation of Second-Order Polynomial Kalman Filter and Falling Object -2

```matlab
XNOISE=SIGNOISE*randn;
X=A0+A1*T+A2*T*T;
XD=A1+2*A2*T;
XDD=2*A2;
XS=X+XNOISE;
RES=XS-XH-TS*XDH-.5*TS*TS*XDDH;
XH=XH+XDH*TS+.5*TS*TS*XDDH+K(1,1)*RES;
XDH=XDH+XDDH*TS+K(2,1)*RES;
XDDH=XDDH+K(3,1)*RES;
SP11=sqrt(P(1,1));
SP22=sqrt(P(2,2));
SP33=sqrt(P(3,3));
XHERR=X-XH;
XDHERR=XD-XDH;
XDDHERR=XDD-XDDH;
SP11P=-SP11;
SP22P=-SP22;
SP33P=-SP33;
count=count+1;
ArrayT(count)=T;
ArrayX(count)=X;
ArrayXH(count)=XH;
ArrayXD(count)=XD;
ArrayXDH(count)=XDH;
ArrayXDD(count)=XDD;
ArrayXHERR(count)=XHERR;
ArraySP11(count)=SP11;
ArraySP11P(count)=SP11P;
ArrayXDHERR(count)=XDHERR;
ArraySP22(count)=SP22;
ArraySP22P(count)=SP22P;
ArrayXDDHERR(count)=XDDHERR;
ArraySP33(count)=SP33;
ArraySP33P(count)=SP33P;
end
```

Fundamentals of Kalman Filtering: A Practical Approach
With Second-Order Filter Altitude Estimate of Falling Object is Near Perfect
It Takes Approximately Ten Sec to Accurately Estimate Velocity of Falling Object With Second-Order Filter
It Takes Nearly Twenty Sec to Accurately Estimate Acceleration of Falling Object With Second-Order Filter
Second-Order Polynomial Kalman Filter Single Flight Results Appear to Match Theory for Errors in Estimate of Altitude
Second-Order Polynomial Kalman Filter Single Flight Results Appear to Match Theory for Errors in Estimate of Velocity
MATLAB Simulation of First-Order Polynomial Kalman Filter and Falling Object-1

ORDER =2;
PHIS=0.;
TS=1;
A0=400000;
A1=-6000.;
A2=-16.1;
XH=0;
XDh=0;
SIGNoise=1000.;
PHI=[1 TS 0 1];
P=[99999999 0 999999999];
IDNP=eye(ORDER);
Q=zeros(ORDER);
H=[1 0];
HT=H';
R=SIGNoise/2;
PHIT=PHI';
Q(1,1)=PHIS*TS^3/3;
Q(1,2)=PHIS*TS*TS/2;
Q(2,1)=Q(1,2);
Q(2,2)=PHIS*TS;
count=0;
for T=0:TS:30

PHIP=PHI*P;
PHIPPHIT=PHIP*PHIT;
M=PHIPPHIT+Q;
MHT=M*HT;
HMHT=H*MHT;
HMHTR=HMHT+R;
HMHTRINV=inv(HMHTR);
K=HMHTRINV*PHIPPHIT;
KH=K*H;
IKH=IDNP-KH;
P=IKH*M;

Fundamentals of Kalman Filtering: A Practical Approach
MATLAB Simulation of First-Order Polynomial Kalman Filter and Falling Object -2

```matlab
XNOISE = SIGNOISE * randn;
X = A0 + A1 * T + A2 * T^2;
XD = A1 + 2 * A2 * T;
XS = X + XNOISE;
RES = XS - XH - TS * XD;
XH = XH + XHERR;
XD = XD + XDHERR;
SP11 = sqrt(P(1,1));
SP22 = sqrt(P(2,2));
XHERR = XH - XH;
XDERR = XD - XD;
SP11P = -SP11;
SP22P = -SP22;
count = count + 1;
ArrayT(count) = T;
ArrayX(count) = X;
ArrayXH(count) = XH;
ArrayXD(count) = XD;
ArrayXDH(count) = XDH;
ArrayXHERR(count) = XHERR;
ArraySP11(count) = SP11;
ArraySP11P(count) = SP11P;
ArraySP22(count) = SP22;
ArraySP22P(count) = SP22P;
end
```

Signal
Kalman filter
Errors in estimates
Save data for plotting and writing to files
First-Order Polynomial Kalman Filter Without Process Noise Cannot Track Second-Order Signal
Adding Process Noise Prevents Altitude Errors of First-Order Filter From Diverging
Adding Process Noise Prevents Velocity Errors of First-Order Filter From Diverging

![Graph showing the comparison between simulation and theory for the error in estimate of velocity. The graph illustrates how adding process noise prevents the velocity errors from diverging. The theory line is shown to be more stable compared to the simulation line.]
Filter Design Making Use of A Priori Information - 1

Model of the real world

\[ \dot{x} = Fx + Gu + w \]

Kalman filter

\[ \hat{x}_k = \Phi_k \hat{x}_{k-1} + G_k u_{k-1} + K_k (z_k - H \Phi_k \hat{x}_{k-1} - HG_k u_{k-1}) \]

Where

\[ G_k = \int_0^{T_s} \Phi(\tau)G \, d\tau \]

In our problem we know that \( \ddot{x} = -g \)

Expressing gravitational information is state space form

\[ \begin{bmatrix} \dot{x} \\ \dot{\dot{x}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} g \]

By inspection

\[ F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \rightarrow \quad \Phi_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \]
Filter Design Making Use of A Priori Information - 2

Recall

\[
\dot{x} = Fx + Gu + w
\]

\[
\begin{bmatrix}
\dot{x} \\
\dot{\hat{x}}
\end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\
\hat{x}
\end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} g
\]

We can also see that

\[
G = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad u = g
\]

Therefore

\[
G_k = \int_0^{T_s} \Phi(\tau)G \ d\tau = \int_0^{T_s} \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} d\tau = \begin{bmatrix} -\frac{T_s^2}{2} \\ -T_s \end{bmatrix}
\]

Since formula of Kalman filter is given by

\[
\hat{x}_k = \Phi_k \hat{x}_{k-1} + G_k u_{k-1} + K_k (z_k - H \Phi_k \hat{x}_{k-1} - H G_k u_{k-1})
\]

Substitution yields

\[
\begin{bmatrix}
\hat{x}_k \\
\hat{\hat{x}}_k
\end{bmatrix} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\
\hat{\hat{x}}_{k-1}
\end{bmatrix} + \begin{bmatrix} -0.5T_s^2 \\ -T_s \end{bmatrix} g + \begin{bmatrix} K_{1s} \\ K_{2s} \end{bmatrix} \begin{bmatrix} x^*_k - [1 \\ 0] \\
[K_{1s} \\ K_{2s}] \begin{bmatrix} \hat{x}_{k-1} \\
\hat{\hat{x}}_{k-1}
\end{bmatrix} - [1 \\ 0] \begin{bmatrix} -0.5T_s^2 \\ -T_s \end{bmatrix} g
\]

Fundamentals of Kalman Filtering: A Practical Approach
Filter Design Making Use of A Priori Information - 3

Multiplying out the terms

\[ \hat{x}_k = \hat{x}_{k-1} + \hat{x}_{k-1}T_s - .5gT_s^2 + K_{1k}(x^*_k - \hat{x}_{k-1} - \hat{x}_{k-1}T_s + .5gT_s^2) \]

\[ \hat{x}_k = \hat{x}_{k-1} - gT_s + K_{2k}(x^*_k - \hat{x}_{k-1} - \hat{x}_{k-1}T_s + .5gT_s^2) \]

If we define the residual as

\[ \text{RES}_k = x^*_k - \hat{x}_{k-1} - \hat{x}_{k-1}T_s + .5gT_s^2 \]

The Kalman filter simplifies to

\[ \hat{x}_k = \hat{x}_{k-1} + \hat{x}_{k-1}T_s - .5gT_s^2 + K_{1k}\text{RES}_k \]

\[ \hat{x}_k = \hat{x}_{k-1} - gT_s + K_{2k}\text{RES}_k \]

Riccati equations are identical to those of first-order filter
MATLAB Version of First-Order Polynomial Kalman Filter with Gravity Compensation-1

\[
\begin{align*}
TS &= 0.1; \\
PHIS &= 0.; \\
A0 &= 40000.; \\
A1 &= 6000.; \\
A2 &= 16.1; \\
XH &= 0.; \\
XD &= 0.; \\
SIGNOISE &= 1000.; \\
ORDER &= 2; \\
T &= 0.; \\
S &= 0.; \\
H &= 0.01; \\
PHI &= \begin{bmatrix} 1 & TS \\ 0 & 1 \end{bmatrix}; \\
P &= \begin{bmatrix} 99999999 & 0 \\ 0 & 999999999 \end{bmatrix}; \\
IDNP &= \text{eye}(ORDER); \\
Q &= \text{zeros}(ORDER); \\
RMAT &= SIGNOISE/2; \\
Q(1,1) &= TS^3 PHIS/3; \\
Q(1,2) &= TS^2 PHIS; \\
Q(2,2) &= PHIS; \\
HMAT &= \begin{bmatrix} 1 & 0 \end{bmatrix}; \\
HT &= HMAT'; \\
PHIT &= PHI'; \\
count &= 0; \\
& \text{for } T = 0:TS:30 \\
& \quad \text{PHIP} = PHI*P; \\
& \quad \text{PHIPPHIT} = PHIP*PHIT; \\
& \quad M = PHIPPHIT + Q; \\
& \quad HM = HMAT*M; \\
& \quad HMHT = HM*HT; \\
& \quad HMHR = HMHT + RMAT; \\
& \quad HMHRINV = \text{inv}(HMHR); \\
& \quad MHT = M*HT; \\
& \quad K = MHT*HMHRINV; \\
& \quad KH = K*HMAT; \\
& \quad KHIDNP = KH + IDNP; \\
& \quad P = KHIDNP*MHAT; \\
\end{align*}
\]

Same fundamental and initial covariance matrices

Same process noise and measurement matrices

Same Riccati equations
MATLAB Version of First-Order Polynomial Kalman Filter with Gravity Compensation-2

XNOISE=SIGNoise*randn;
X=A0+A1*T+A2*T*T;
XD=A1+2*A2*T;
XS=X+XNOISE;
RES=XS-XH-TS*XDH+16.1*TS*TS;
XH=XH+XDH*TS-16.1*TS*TS+K(1,1)*RES;
XD=XDH-XD-32.2*TS+K(2,1)*RES;
SP11=sqrt(P(1,1));
SP22=sqrt(P(2,2));
XHERR=X-XH;
XDERR=XD-XD;
SP11P=-SP11;
SP22P=-SP22;
count=count+1;
ArrayT(count)=T;
ArrayX(count)=X;
ArrayXH(count)=XH;
ArrayXD(count)=XD;
ArrayXDH(count)=XD;
ArrayXHERR(count)=XHERR;
ArraySP11(count)=SP11;
ArraySP11P(count)=SP11P;
ArrayXDERR(count)=XDERR;
ArraySP22(count)=SP22;
ArraySP22P(count)=SP22P;
end

Filter now has gravity compensation
New First-Order Filter Without Process Noise Reduces Altitude Errors From That of a Second-Order Filter Without Process Noise
## Estimation Accuracy After Thirty Sec of Track

<table>
<thead>
<tr>
<th>Filter</th>
<th>Position Error</th>
<th>Velocity Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second-Order (No Process Noise)</td>
<td>200 ft</td>
<td>25 ft/sec</td>
</tr>
<tr>
<td>First-Order (Process Noise)</td>
<td>300 ft</td>
<td>150 ft/sec</td>
</tr>
<tr>
<td>First-Order Gravity Compensation (No Process Noise)</td>
<td>100 ft</td>
<td>5 ft/sec</td>
</tr>
</tbody>
</table>
Revisiting Accelerometer Testing Example
**Accelerometer Experiment Test Setup**

![Diagram of accelerometer experiment test setup]

**Developing equations**

\[
\text{Accelerometer Output} = g\cos\theta_k + B + SFg\cos\theta_k + K(g\cos\theta_k)^2
\]

Theory \( = g\cos\theta_k \)

Error \( = \text{Accelerometer Output} - \text{Theory} = B + SFg\cos\theta_k + K(g\cos\theta_k)^2 \)

**If measurement of angle contaminated with noise**

\[
\text{Error} = \text{Accelerometer Output} - \text{Theory} = g\cos\theta_K^* + B + SFg\cos\theta_K^* + K(g\cos\theta_K^*)^2 - g\cos\theta_K
\]
# Nominal Values For Accelerometer Testing Example

<table>
<thead>
<tr>
<th>Term</th>
<th>Scientific Value</th>
<th>English Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias Error</td>
<td>10 μg</td>
<td>$10 \times 10^{-6} \times 32.2 = 0.000322 \text{ ft/sec}^2$</td>
</tr>
<tr>
<td>Scale Factor Error</td>
<td>5 ppm</td>
<td>$5 \times 10^{-6}$</td>
</tr>
<tr>
<td>G-Squared Sensitive Drift</td>
<td>1 μg/g$^2$</td>
<td>$1 \times 10^{-6} / 32.2 = 3.106 \times 10^{-8} \text{ sec}^2/\text{ft}$</td>
</tr>
</tbody>
</table>
Kalman Filter Formulation - 1

If bias, scale factor and g-sensitive drift are constant

\[
\begin{bmatrix}
\dot{B} \\
\dot{SF} \\
\dot{K}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
B \\
SF \\
K
\end{bmatrix}
\]

We are neglecting process noise

Therefore systems dynamics matrix is zero

\[
F = 
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

Which means

\[
\Phi_k = I + FT_s + ... = 
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix} + 
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}T_s = 
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

If we assume that the measurement is the previously defined error

\[
z_k = g\cos\theta_K^* + B + SFg\cos\theta_K^* + K(g\cos\theta_K^*)^2 - g\cos\theta_K
\]

We can develop a measurement equation

\[
z_k = 
\begin{bmatrix}
1 & g\cos\theta_K^* & (g\cos\theta_K^*)^2 \\
B & SF \\
K
\end{bmatrix}
\] + \nu_k
Therefore the measurement matrix is

\[ H = \begin{bmatrix} 1 & g \cos \theta^*_K & (g \cos \theta^*_K)^2 \end{bmatrix} \]

The equivalent measurement noise is

\[ v_k = g \cos \theta^*_K - g \cos \theta_K = g(\cos \theta^*_K - \cos \theta_K) \]

since the real measurement noise is the noise on the angle
Finding Measurement Noise Matrix For Riccati Equations

Recall

\[ v_k = g \cos \theta_k^* - g \cos \theta_k = g (\cos \theta_K^* - \cos \theta_K) \]

Assume noisy angle is true angle plus a small term

\[ v_k = g [\cos(\theta_K + \Delta \theta_k) - \cos \theta_K] \]

Using trigonometric expansion

\[ \cos(\theta_K + \Delta \theta_k) = \cos \theta_k \cos \Delta \theta_k - \sin \theta_k \sin \Delta \theta_k \]

And making small angle approximation

\[ v_k = g [\cos \theta_k \cos \Delta \theta_k - \sin \theta_k \sin \Delta \theta_k - \cos \theta_K] \approx -g \Delta \theta_k \sin \theta_k \]

Squaring and taking expectations yields

\[ R_k = E(v_k^2) = (g \sin \theta_k)^2 E(\Delta \theta_k^2) = g^2 \sin^2 \theta_k \sigma_\theta^2 \]

To be used in Riccati equation
Equations for Kalman Filter

General formula

\[ \hat{x}_k = \Phi_k \hat{x}_{k-1} + K_k (z_k - H \Phi_k \hat{x}_{k-1}) \]

Substitution yields

\[
\begin{bmatrix}
\hat{B}_k \\
\hat{SF}_k \\
\hat{K}_k
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\hat{B}_{k-1} \\
\hat{SF}_{k-1} \\
\hat{K}_{k-1}
\end{bmatrix} +
\begin{bmatrix}
K_{1k} \\
K_{2k} \\
K_{3k}
\end{bmatrix}
\begin{bmatrix}
z_k - 1 \\
g \cos \theta_k^* \\
(g \cos \theta_k^*)^2
\end{bmatrix}
\]

Multiplying out the equations and defining a residual

\[ \text{RES}_k = z_k - \text{BIAS}_{k-1} - \hat{SF}_{k-1} g \cos \theta_k^* - \hat{K}_{k-1} (g \cos \theta_k^*)^2 \]

\[ \hat{B}_k = \hat{B}_{k-1} + K_{1k} \text{RES}_k \]

\[ \hat{SF}_k = \hat{SF}_{k-1} + K_{2k} \text{RES}_k \]

\[ \hat{K}_k = \hat{K}_{k-1} + K_{3k} \text{RES}_k \]
MATLAB Simulation of Accelerometer Filter-1

```
ORDER=3;
BIAS=.00001*32.2;
SF=.000005;
XK=.000001/32.2;
SIGTH=.000001;
G=32.2;
BIASH=0.;
SFH=0.;
XKH=0;
SIGNoise=.000001;
S=0.;
Q=zeros(ORDER);
PHI=[1 0 0; 0 1 0;0 0 1];
IDNP=eye(ORDER);
PHIT=PHI';
P=[9999999990 0 0 9999999990 0 0 9999999999999];
count=0;
for THETDEG=0:2:180
    THET=THETDEG/57.3;
    THETNOISE=SIGNoise*randn;
    THETS=THET+THETNOISE;
    HMAT=[1 G*cos(THETS) (G*cos(THETS))^2];
    HT=HMAT';
    R=(G*sin(THETS)*SIGTH)^2;
    PHIP=PHI*P;
    PHIPPHIT=PHIP*PHIT;
    M=PHIPPHIT+Q;
    HM=HMAT*M;
    HMHT=HM*HT;
    HMHTR=HMHT+R;
    HMHTRINV=inv(HMHTR);
    MHT=M*HT;
    K=MHT*HMHTRINV;
    KH=K*HMAT;
    IKH=IDNP-KH;
    P=IKH*M;
```
Z = BIAS + SF * G * cos(THETS) + XK * (G * cos(THETS)^2 - G * cos(THET)) + G * cos(THETS);  
RES = Z - BIAS - SF * H * G * cos(THETS) - XKH * (G * cos(THETS)^2);  
BIAS = BIAS + K(1,1) * RES;  
SFH = SFH + K(2,1) * RES;  
XKH = XKH + K(3,1) * RES;  
SP11 = sqrt(P(1,1));  
SP22 = sqrt(P(2,2));  
SP33 = sqrt(P(3,3));  
SP11P = SP11;  
SP22P = SP22;  
SP33P = SP33;  
BIASERR = BIAS - BIA SH;  
SFERR = SF - SFH;  
XKERR = XK - XKH;  
ACTNOISE = G * cos(THETS) - G * cos(THET);  
SIGR = sqrt(R(1,1));  
SIGRP = SIGR;  
count = count + 1;  
ArrayTHETDEG(count) = THETDEG;  
ArrayBIAS(count) = BIAS;  
ArrayBIASH(count) = BIA SH;  
ArraySF(count) = SF;  
ArraySFH(count) = SFH;  
ArrayXK(count) = XK;  
ArrayXKH(count) = XKH;  
ArrayBIASERR(count) = BIASERR;  
ArraySP11(count) = SP11;  
ArraySP11P(count) = SP11P;  
ArraySFERR(count) = SFERR;  
ArraySP22(count) = SP22;  
ArraySP22P(count) = SP22P;  
ArrayXKERR(count) = XKERR;  
ArraySP33(count) = SP33;  
ArraySP33P(count) = SP33P;  
ArrayACTNOISE(count) = ACTNOISE;  
ArraySIGR(count) = SIGR;  
ArraySIGRP(count) = SIGRP;
Derived Formula for Standard Deviation of Equivalent Noise Appears to be Correct

\[ \sigma_\theta = 1 \mu R \]

Simulated and theoretical results are shown in the graph.

- **Simulation**
- **Theory**
Kalman Filter Appears to be Working Correctly Because Actual Error in Estimate of Accelerometer Bias is Within Theoretical Bounds

![Graph showing comparison between Simulation and Theory for error in estimate of bias with measurement angle. The graph includes error bars and theoretical curves.](image)
Kalman Filter Appears to be Working Correctly Because Actual Error in Estimate of Accelerometer Scale Factor is Within Theoretical Bounds

![Graph showing comparison of simulation and theory for error in estimate of scale factor against measurement angle. The graph includes a vertical axis labeled "Error in Estimate of Scale Factor" and a horizontal axis labeled "Measurement Angle (Deg)." The graph shows a comparison between simulation data (red bars) and theoretical data (blue line) for different values of measurement angle. The theoretical line is labeled "Theory" and the simulation data is labeled "Simulation." The graph indicates that the theoretical and simulation data match closely within the bounds of theoretical error.]
Kalman Filter Appears to be Working Correctly Because Actual Error in Estimate of Accelerometer G-Sensitive Drift is Within Theoretical Bounds
Monte Carlo Experiments Suggest Kalman Filter Estimates Accelerometer Bias Accurately

![Graph showing Monte Carlo experiments indicating accurate estimates of accelerometer bias. The graph plots bias in units of μg² versus run number, with estimates for σ₀ = 1 μR and σ₀ = 10 μR, and actual values as well.](image_url)
Monte Carlo Experiments Suggest Kalman Filter Estimates Accelerometer Scale Factor Error Accurately

![Graph showing the comparison between estimate and actual scale factor error with different standard deviations.]

- Estimate \( \sigma_\theta = 1 \mu R \)
- Estimate \( \sigma_\theta = 10 \mu R \)
- Actual

Scale Factor

Run Number

7x10^{-6}

6
5
4
3
2
1
0
25
20
15
10
5
0

Fundamentals of Kalman Filtering: A Practical Approach
Monte Carlo Experiments Suggest Kalman Filter Estimates Accelerometer G-Sensitive Drift Accurately
Monte Carlo Results Indicate That Kalman Filter is Not Able to Estimate Accelerometer Bias When There is 100 $\mu R$ of Measurement Noise
Monte Carlo Results Indicate That Kalman Filter is Not Able to Estimate Accelerometer Scale Factor Error When There is 100 µR of Measurement Noise
Monte Carlo Results Indicate That Kalman Filter is Not Able to Estimate Accelerometer G-Sensitive Drift When There is 100 µR of Measurement Noise
Polynomial Kalman Filters
Summary

• Polynomial Kalman filtering equations presented
• Showed equivalence between polynomial Kalman filter without process noise and least squares recursive filter
• Numerical example presented showing filtering options for tracking a falling object
  - Showed a priori information can be useful
• Designed polynomial Kalman filter for accelerometer testing problem