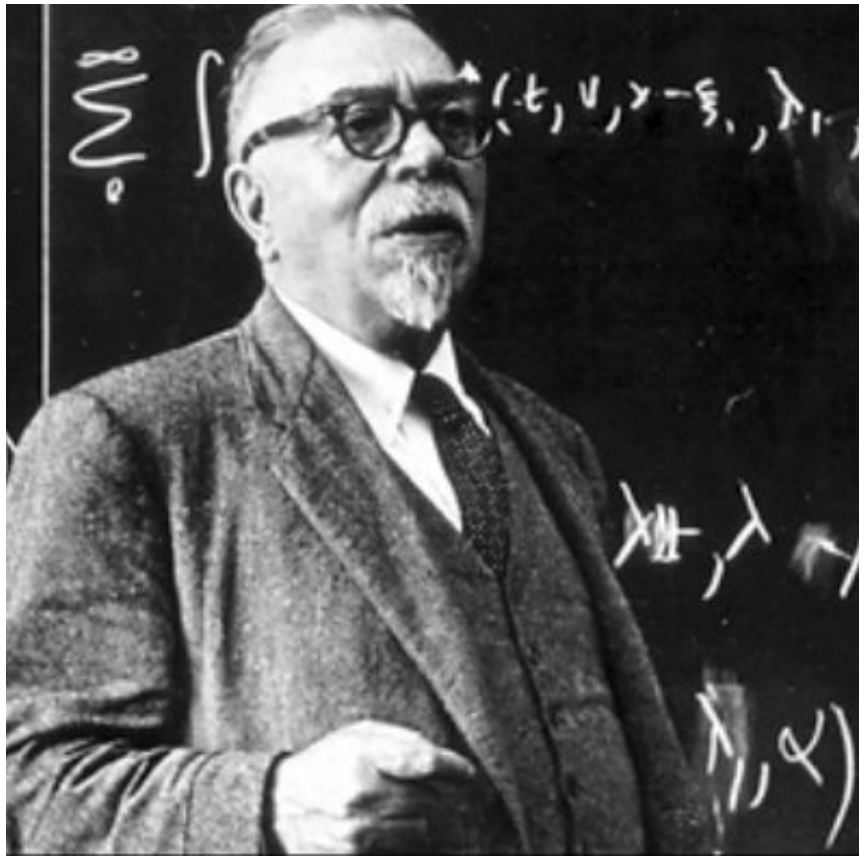
# Kalman Filters

T.J.Moir


positivit
The



# Kalman Filter is a State Estimator

- The Kalman filter is like an Observer ie from measurements of the output of a system and the dynamic model of the system we can get the best least-squares estimate of the state. We use the state in a state-feedback control law. Other disciplines may use a Kalman filter for filtering of images.
- Kalman filter works on noisy signals but we must know the statistical properties of the noise. This is one disadvantage.
- Previous work was the work of Norbert Wiener with the Wiener filter which is similar in some respects but is restricted to systems which are time-invariant and have stationary noise. KF can be used even if the noise statistics changes or the system is time-varying. Of course these changes must be known.

# Continuous-time state-space model – stochastic, proper system

$$\dot{x}(t) = Ax(t) + Bu(t) + D\omega(t)$$

$$y(t) = Cx(t) + V(t)$$

$\omega(t), V(t)$ Are column noise vectors – for our application they have dimension 2 and represent driving noise (process noise) and measurement noise respectively. They are assumed to be white, have zero dc in them (zero-mean) and have covariance matrices as follows. Note that D is used in a different Way from normal. Usually D is used as an output matrix – different notation here.

$$E\left[\omega(t)\omega^T(t)\right] = Q \quad E\left[V(t)V^T(t)\right] = R$$

Both matrices must be positive definite ie $Q > 0, R > 0$

# What does positive-definite mean?

- For scalar quantities say a number 3, we say 3>0 or if it was negative
- -3<0. Likewise we could say some term a>=0. Greater than equal to zero. But with matrices this needs extending to A>0 positive definite, A<0 negative definite and A>=0 positive semidefinite (also negative semi-definite). For example

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} > 0 \qquad a_{11} > 0, \left|A\right| > 0$$

$$a_{11} > 0, a_{11}a_{22} - a_{12}a_{21} > 0$$

# Discrete-time stochastic system. We no longer bother with continuous time Kalman filters though they were the first to be derived and used. Sampling interval T seconds

$$x_{k+1} = Fx_k + \Delta u_k + G\omega_k$$

$$y_k = Hx_k + V_k$$

$$F = e^{AT} \quad \Delta = \int_0^T e^{A\sigma} B d\sigma \quad G = \int_0^T e^{A\sigma} D d\sigma$$

These matrices can be found using Matlab or Laplace Transforms. Matlab uses numerical methods eg a power series

$$F = e^{AT} = I + AT + A^2T^2 / 2! + \ldots$$

# Matlab commands (J is zero usually, fs=10kHz)

$$J = [0];$$
$$sys = ss(A, B, C, J)$$
$$\mathrm{T} = 0.0001$$
$$sysd = c2d(sys, T)$$

# Example, double integrator.

$$G(s) = \frac{1}{s^2}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

States here have physical meaning, position and velocity, x1 and x2.

In Matlab A=[0 1;0 0],B=[0 1]’; C=[1 0]; J=[0];

```
sys=ss(A,B,C,J)

sys =

 A =
     x1  x2
  x1   0   1
  x2   0   0

 B =
     u1
  x1   0
  x2   1

 C =
     x1  x2
  y1   1   0

 D =
     u1
  y1   0

Continuous-time state-space model.
```

```
sysd=c2d(sys,0.0001)

sysd =

 A =
        x1      x2
  x1       1  0.0001
  x2       0       1

 B =
        u1
  x1   5e-09
  x2  0.0001

 C =
     x1  x2
  y1   1   0

 D =
     u1
  y1   0

Sample time: 0.0001 seconds
Discrete-time state-space model.
```

$$\begin{bmatrix} x^1_{k+1} \\ x^2_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \Delta = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}, H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Kalman filter is actually a one-step ahead predictor given by

$$\hat{x}_{k+1/k} = F\hat{x}_{k/k-1} + Ke_k + \Delta u_k$$

$$e_k = y_k - H\hat{x}_{k/k-1}$$

$$K = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

The delta matrix may not always exist. Depends if it is a control problem or just an estimation problem. Usually u is deterministic and the control signal to the system under control. For example, filtering a random signal from white noise has no control term – obviously.
But how do we find the K vector which is known as the Kalman Gain matrix? In this case K is a vector because there is only one input and output.

Calculation of the Kalman Gain vector (or matrix) K. This matrix is in turn dependent on another matrix P known as the error covariance matrix of the state. P needs to be initialised to a diagonal matrix (say 10xI where I is the identity matrix). P is found from a Riccati equation

Loop forever – real-time loop
{

$$K_k = FP_k H^T (R + HP_k H^T)^{-1}$$

$$P_{k+1} = FP_k F^T + GQG^T - K_k HP_k F^T$$

}

The above two equations can be done at the same time as the filter itself, but if the system dynamics do not change with time and the noise statistics do not change there is little point. This can then be done offline. K will converge to constant values typically after a few hundred iterations.

Alternative form

Predict

$$\hat{x}_{k/k-1} = F\hat{x}_{k-1/k-1} + \Delta u_k$$

$$P_{k/k-1} = FP_{k-1/k-1}F^T + GQG^T$$

Filter based on state-predictor (gives slightly smaller mean-square error)

$$\hat{x}_{k/k} = \hat{x}_{k/k-1} + K_k^f \left( y_k - H\hat{x}_{k/k-1} \right)$$

$$K_k^f = P_{k/k-1}H^T (R + HP_{k/k-1}H^T)^{-1}$$

$$P_{k/k} = P_{k/k-1} - KHP_{k/k-1}$$

# Real example. 6 axis Gyro and accelerometer

Many applications need to know the orientation of a system in space. For example a rocket, Satellite, drone etc. For inexpensive but less accurate solutions, various MEMS devices are available which have built in Gyros for each of 3 axis and accelerometers for 3 axis too. Here we are only interested in the pitch for a Segway type device that needs to measure tilt angle.

The rate gyro measures velocity in each axis – x,y,z and therefore to get position we need to integrate this measurement. The trouble is that gyro measurements drift over time and have a bias. The accelerometer can also measure angle using the arctan function but it is very sensitive to high frequency vibrations and so it too has difficulties, but it doesn't drift. Therefore a system is needed which combines the two reading in what we call sensor fusion. Denoting the three accelerometer outputs as Rx,Ry and Rz, then the angle for pitch is

$$\theta_p = \tan^{-1}\left(\frac{R_y}{\sqrt{R_z{}^2 + R_x{}^2}}\right)$$

# State-variable Model. Two inputs, two outputs.

$$\begin{bmatrix} \dot{\theta}_p \\ \dot{\omega}_p \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_p \\ \omega_p \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_g \\ \xi_a \end{bmatrix}$$

$$\begin{bmatrix} \omega_p \\ \theta_p \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_p \\ \omega_p \end{bmatrix} + \begin{bmatrix} v_g \\ v_a \end{bmatrix} = \begin{bmatrix} Gyro\_out + noise1 \\ AccelAngle\_out + noise2 \end{bmatrix}$$

$\xi_g$ and $\xi_a$ represent the random *inputs* to the Gyro and Accelerometer respectively i.e. velocity and acceleration
$v_g$ and $v_a$ represent the random *measurement* noise to the Gyro and Accelerometer measurements

The minus sign is because of ***negative acceleration*** due to gravity.
Discretising with sample interval T gives us

$$-\frac{1}{s^2}$$

# Discrete-Time Model of Gyro and Accelerometer

$$\begin{bmatrix} \theta^p_{k+1} \\ \omega^p_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & -T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta^p_k \\ \omega^p_k \end{bmatrix} + \begin{bmatrix} T & -T^2/2 \\ 0 & T \end{bmatrix} \begin{bmatrix} \xi_g \\ \xi_a \end{bmatrix}$$

$$\begin{bmatrix} \omega^p_k \\ \theta^p_k \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \theta^p{}_k \\ \omega^p{}_k \end{bmatrix} + \begin{bmatrix} \nu_g \\ \nu_a \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & -T \\ 0 & 1 \end{bmatrix}, G = \begin{bmatrix} T & -T^2/2 \\ 0 & T \end{bmatrix}, H = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

# Find the analogue Kalman filter first

Use Matlab with our analogue system.

$$A=\begin{bmatrix}0 & -1\\ 0 & 0\end{bmatrix}, D=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}, C=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

Arbitrarily choose $$Q=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}, R=\begin{bmatrix}0.001 & 0\\ 0 & 0.001\end{bmatrix}$$

This will give us an idea of what kind of filter emerges from the Kalman approach

# Matlab

```
>> A=[0 -1
0 0]

A =

    0    -1
    0     0

>> B=eye(2)

B =

    1     0
    0     1

>> C=[0 1
1 0]

C =

    0     1
    1     0
```

```
>> sys=ss(A,B,C,D)

sys =

  A =
       x1  x2
   x1   0  -1
   x2   0   0

  B =
       u1  u2
   x1   1   0
   x2   0   1

  C =
       x1  x2
   y1   0   1
   y2   1   0

Continuous-time state-space model.
```

```
>> tf(sys)

ans =

  From input 1 to output...
   1:  0

       1
   2:  -
       s

  From input 2 to output...
       1
   1:  -
       s

      -1
   2:  ---
       s^2
```

# Transfer function matrix of combined Gyro and accelerometer tf(sys)

$$A=\begin{bmatrix}0 & -1\\ 0 & 0\end{bmatrix}, D=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}, C=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

$$\begin{bmatrix}\omega_p(s)\\ \theta_p(s)\end{bmatrix}=\begin{bmatrix}0 & 1/s\\ 1/s & -1/s^2\end{bmatrix}\begin{bmatrix}\xi_g(s)\\ \xi_a(s)\end{bmatrix}$$

# Analogue steady-state Kalman Filter

[kest,K,P]=kalman(sys,Q,R,0)

K =

-0.4999 31.6346
31.6188 -0.4999

P =

0.0316 -0.0005
-0.0005 0.0316

The transfer function matrix of the Kalman filter is found from

$$G_k(s) = C\left(sI - (A - KC\right)^{-1} K$$

$$= \frac{1}{s^2 + 63.2s + 1000}\begin{bmatrix} 31.6s + 100 & -0.5s \\ -(0.5s + 31.6) & 31.6s + 100 \end{bmatrix}$$

# Examining the Kalman Filter transfer functions

From the transfer functions of the transfer function matrix we can easily deduce what the filter is doing. It's two outputs are performing

Position Estimate output= Low-pass filter Gyro output – Band pass filter Accelerometer

Velocity Estimate output= - Low-pass filter Gyro output + Low pass filter Accelerometer

# Software. Loop forever in real-time.

{

$$\hat{x}_{k+1/k} = F\hat{x}_{k/k-1} + Ke_k$$

$$e_k = y_k - H\hat{x}_{k/k-1}$$

}

Iterate the above algorithm in real-time. The Kalman gain matrix need not be updated. Instead we use the steady-state Kalman gain found from the solution of the steady-state Riccati equation – offline!

The only time we need update the K matrix is when the statistics change with time (non-stationary) or if the system is time-variant.