LQG controller design example

Example of simple vertical motion of a drone (Altitude control problem).

state $\rightarrow$ $x_k = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$

Input $\rightarrow$ u_k : control force / thrust command

output $\rightarrow$ y_k : position measurement + measurement noise

The discrete-time linear model :

$$x_{k+1} = A x_k + B u_k + w_k$$

$$y_k = C x_k + v_k$$

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \qquad \Delta t = 0.1 \text{ second}$$

$$A = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}$$

position

$$x_{K+1} \rightarrow \begin{bmatrix} p \\ v \end{bmatrix}_{K+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix}_K + Bu_K + w_K$$

velocity

$$B = \begin{bmatrix} 0.005 \\ 0.1 \end{bmatrix}$$

$\frac{1}{2}\Delta t^2$

Δt

$$y_K \text{ (measurement)} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

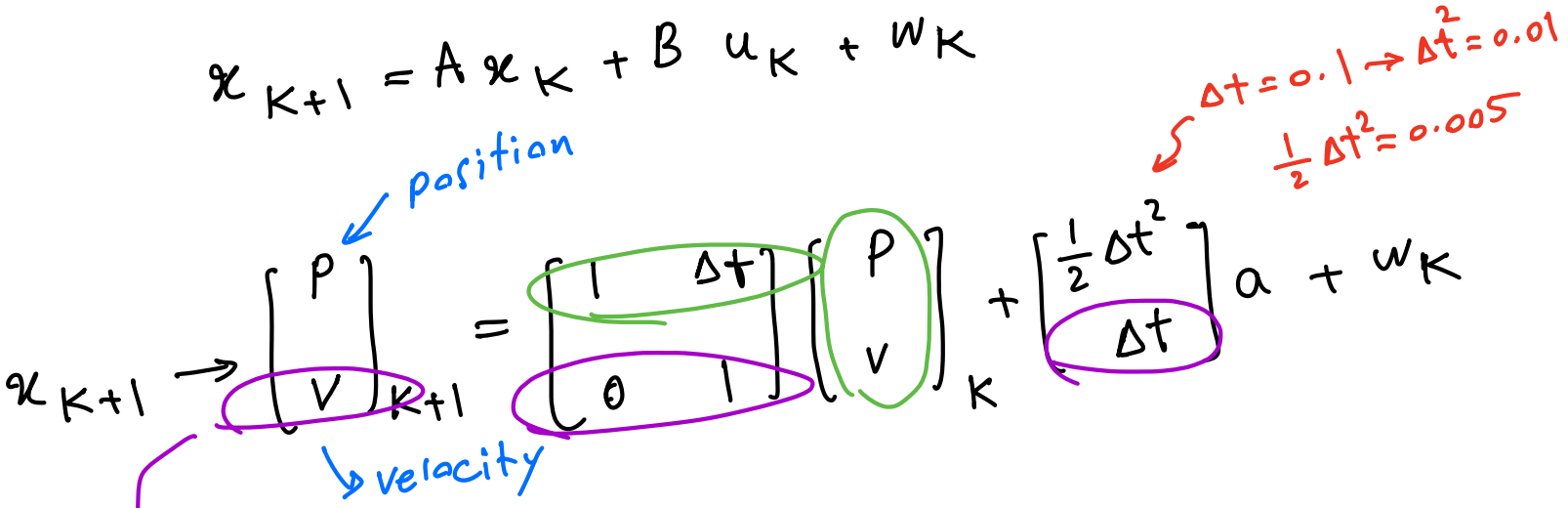
measuring position in this example (e.g. using GPS or barometer)

sensor used for vertical motion measurement

u_K is the control input which affects the acceleration of the vehicle ($\Sigma F = ma$)

therefor, u_k is acceleration.

$$x_{k+1} = A x_k + B u_k + w_k$$



position = initial position

$$p_{k+1} = 1 \times p_k + \Delta t\, v_k + \frac{1}{2} a \Delta t^2$$

velocity

acceleration

$$x = x_0 + v_0 t + 1/2\, a t^2$$

same kinematics equation as in dynamics

$$v_{k+1} = v_k + \Delta t\, a$$

Same kinematics equation in dynamics:

$$a = \frac{\Delta v}{\Delta t} = \frac{(v_{k+1}) - v_k}{\Delta t} \Rightarrow v_{k+1} = v_k + a \Delta t$$

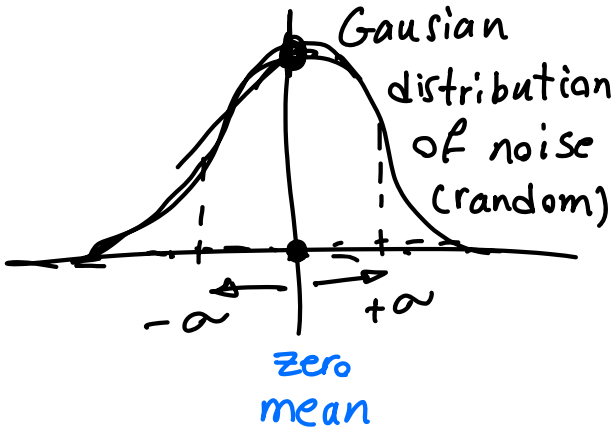
The process (mathematical predicted model) noise and the measurement (from the sensor) noise

are gausian:

process noise:

$$w_k \sim N(0, W)$$

mean of zero



measurement noise:

$$v_k \sim N(0, V)$$

These values are given as below for this example:

$$W = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$$

process noise
(uncertainty in the model)

The model has to be tested experimentally to verify the uncertainty in the model.

$$V = \begin{bmatrix} 0.1 \end{bmatrix}$$

uncertainty in sensor measurement
(measurement Noise)

From the sensar datasheet.

LQR Design:

LQG → LQR + Kalman Filter

LQR: optimal control

Kalman Filter: Deals with uncertainties

LQR algorithm $\xrightarrow{\text{gives}}$ K optimal control gain

$u = K x$

control force ↓ LQR control gain

x → states of the system.

LQG algorithm

$u = K \hat{x}$

control force ↓ same LQR control gain

$\hat{x}$ → The estimated state obtained from Kalman Filter

LQR Design:

we want to minimize the cost:

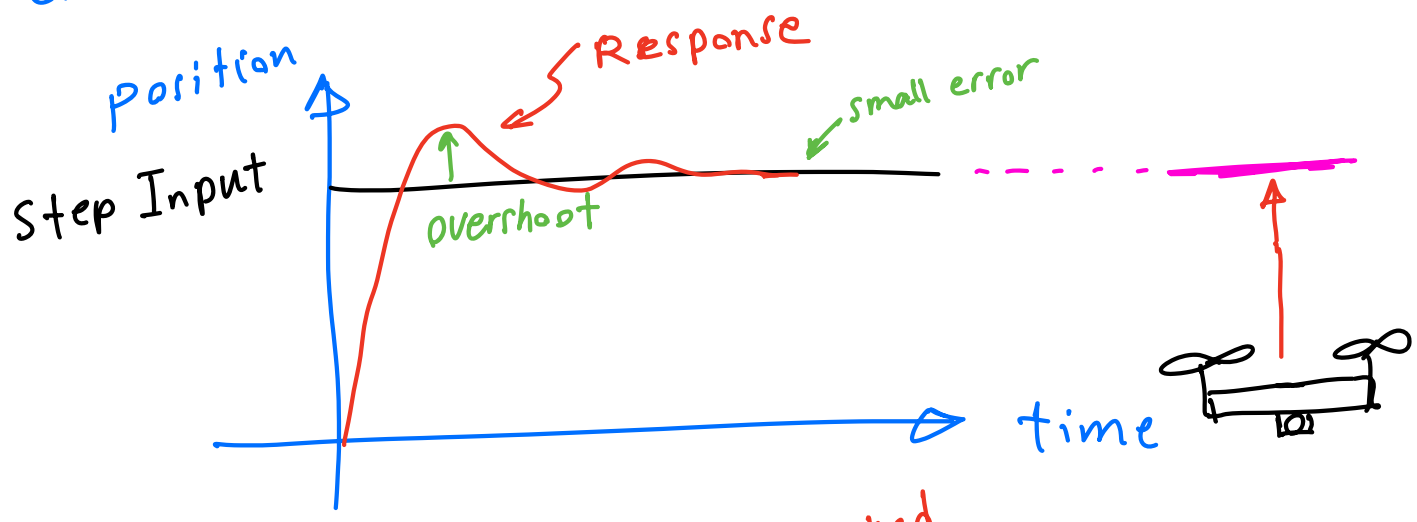
$$J = \sum (x_k^T Q x_k + u_k^T R u_k)$$

J ← cost function

x_k ↓ state (Response of the system)

u_k ↓ control effort by the actuators

Minimizing the effort and minimizing the error in the response.



Define:

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$$

10 multiplied by position

multiplied by velocity

we assigned 10 for position error (relative to 1 for velocity error), the error in position is very important for us (10 is a much larger compared to 1 for velocity). so we penalize the position error by 10 and penalize the velocity error by 1.

Define:

$$R = 1$$

Penalizing control effort by value of 1.

In order to find the LQR control gain "K", we need to solve the Descrete Algebraic Riccati Equation (DARE):

$$P = A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A + Q$$

Gain $$K = (R + B^T P B)^{-1} B^T P A$$

P is LQR Riccati matrix

K is the optimal LQR feedback gain

using MATLAB to find "K"

```
dt = 0.1;   % time step

A = [1  dt;
     0   1];

B = [0.5*dt^2;
     dt];

Q = [10  0;
     0   1];
```

```
R = 1;
```

Solve for DARE:

```
[P, ~, ~] = dare(A, B, Q, R)
```

we need P

~ means we don't need these results.

```
K = (R + B' * P * B) \ (B' * P * A);
```

or we could use the "dlqr" command to get the gain K as below:

```
[K_dlqr, P_dlqr, ~] = dlqr(A, B, Q, R)
```

LQR gain K

Kalman Filter Design

$$y_k = Cx_k + v_k$$

$$\hat{x}_k = A\hat{x}_k + Bu_k$$

$$P_{K+1} = A P_K A^T + W$$

$$S_{K+1} = C P_{K+1} C^T + V$$

Kalman Gain : $L_{K+1} = P_{K+1|K} C^T S^{-1}_{K+1}$

state update :

$$\hat{x}_{K+1|K+1} = \hat{x}_{K+1|K} + L_{K+1}(Y_{K+1} - C\hat{x}_{K+1|K})$$

Using MATLAB to calculate the kalman gain and the state estimate :

```
dt = 0.1;          % ← time step

A = [1   dt;
     0   1];

B = [0.5 * dt^2;
     dt];

C = [1   0];
```

process noise covariance (W) and measurement nois covariance (V)

```
W = [0.01   0;
     0    0.01];

V = 0.1;        (single measurement, therefore one value only)

[Pe, ~, ~] = dare(A', C', W, V);
```

↑ error covariance matrix

The Kalman gain:

```
L = A * Pe * C' / (C * Pe * C' + V);
```

↑ Kalman gain

Note: This Kalman gain (for LQG) will remain the same value throughout the calculations in the LQG Control algorithm.
(Note: in Kalman filter that is used for position estimation only (not LQG) Kalman gain is updated in each step of calculation to give you the best estimate).

Alternatively, we could calculate the Kalman gain directly in one step using the following MATLAB command:

```
KLQG = lqg(sys, W, V)
```

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