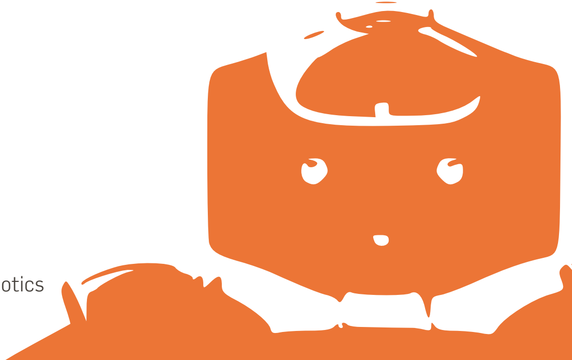
You can download the sources of this presentation here:
github.com/severin-lemaignan/module-mobile-and-humanoid-robots

ROBOTICS
WITH
PLYMOUTH
UNIVERSITY

ROCO318
Mobile and Humanoid Robots
Part 3 - Kalman filters



Séverin Lemaignan

Centre for Neural Systems and Robotics
Plymouth University

PART 3 – KALMAN FILTERS

For further reading, see:

- Welch and Bishop (2001) An introduction to the Kalman filter, SIGGRAPH 2001, ACM.
- Autonomous Mobile Robots, chapter 5.6.8

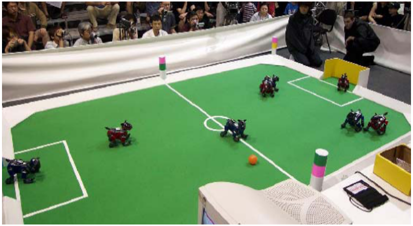
WHAT IS A KALMAN FILTER?

For example: speed, height, position, acceleration, ...

- Developed by Rudolph E. Kalman in 1960.
- Mathematical tool that estimates the real **state** of a system based on uncertain sensor readings.
- It assumes the system is **linear** and noise is **normal** (aka Gaussian).
- Gives past, present and **future estimations**.
- Still very effective and useful for all other classes of systems.
- Hugely popular in digital control systems.

APPLICATIONS OF KALMAN FILTERS

- Estimating critical flight parameters for guidance of missiles.
- Sensor fusion in aircraft.
- Fusion of localisation estimates in GPS.
- Estimating game controller sensor information.
- Prediction of ball position in robot football.
- Prediction of head and hands position and orientation in 3D body posture capture system.
- Prediction of the stock market.
- ...



SOME KALMAN FILTER FACTS

It is a filter? Not really, it does more than filters do

- Taking into account sensor measurements and process variables.
- Prediction forwards (and backwards if needed) in time.
- No explicit frequency response

Kalman Filter is **recursive**

- It start with initial estimates and continuously updates these estimates according to the process model and sensor measurements coming in.

Highly efficient: **Polynomial in measurement dimensionality** k and **state dimensionality** n: $O(k^{2.376} + n^2)$

Optimal, i.e. there is no way of doing better.

BASIC CONCEPTS

STATE

The **state** of a process is a vector of real numbers capturing the relevant information describing the process. $\mathbf{x} = \mathbb{R}^n$

For example

- The position and speed of a wheeled robot: $\mathbf{x} = [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}]$
- The speed of a missile: $\mathbf{x} = [v, t_{thurst}]$

STOCHASTICITY

The true state of a system is unknown

- We don't know the true speed of an aircraft, or the true location of the robot.

This is due to **stochastic** (= random) noise in the measurements and the process.

- Measurement noise example: the air pressure meter reading fluctuates, even at the same altitude.
- Process noise example: even if we keep the accelerator in the same position the car never goes at exactly 60 mph.

LINEAR SYSTEM

A linear equation is a sum of input variables

- For example $f(x) = 5x + 3$ is linear, $f(x) = cos(x)$ is not.
- A linear system can be written in matrix form as $\mathbf{y} = A \cdot \mathbf{x}$ or

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

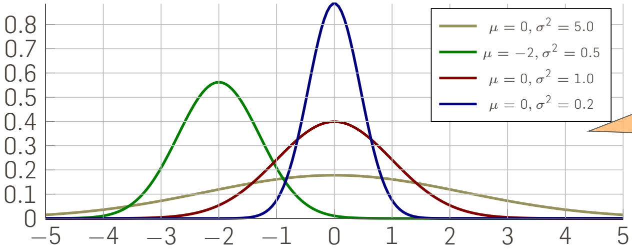
NORMAL DISTRIBUTION

Normal or Gaussian

- Symmetrical distribution, captured with two values: **mean** μ **and variance** σ^2.
- Described by:

$$\varphi_{\mu,\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

This factor keeps the integral (surface under the curve) equal to 1



If these would be measurement distributions of sensors, which sensor is the best?

NORMAL DISTRIBUTION (2)

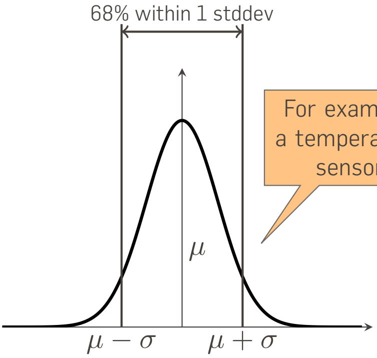
The Kalman Filter assumes that **measurement and process noise are normal** (also known as Gaussian) and **independent**.

Univariate

$$p(x) \sim \mathcal{N}(\mu, \sigma^2):$$

Probability distribution

$$p(x) \sim \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



For example, a temperature sensor

NORMAL DISTRIBUTION (2)

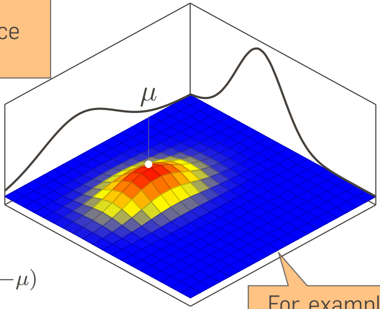
The Kalman Filter assumes that **measurement and process noise are normal** (also known as Gaussian) and **independent**.

Multivariate

$n \times n$ covariance matrix

$$p(\mathbf{x}) = p\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) \sim \mathcal{N}_n(\mu, \Sigma) :$$

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x_n - \mu)^\top \Sigma^{-1} (x_n - \mu)}$$



For example, the x and y position of a robot

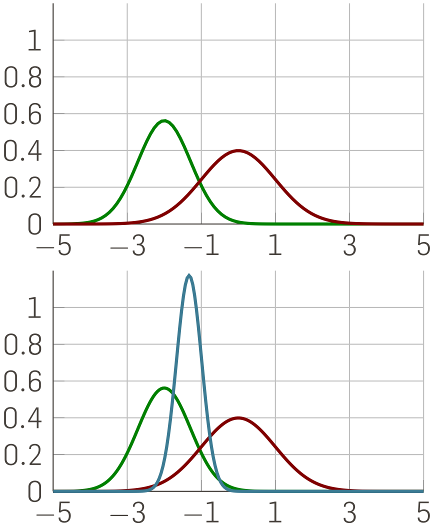
PRODUCT OF NORMAL DISTRIBUTIONS

Combining Gaussians (μ, σ^2) and (ν, r^2):

$$\mu' = \frac{1}{\sigma^2 + r^2}(r^2\mu + \sigma^2\nu)$$

$$\sigma^{2\prime} = \frac{1}{\frac{1}{\sigma^2} + \frac{1}{r^2}}$$

Combining two Gaussians results in a Gaussian that has a ***smaller standard deviation***.



COVARIANCE

Covariance: measure of how two variables change together.

Two series X and Y of values, each of size n.

$$cov(X,Y) = \overline{(X-\bar{X})(Y-\bar{Y})} = \sum_{i=1}^{n} \frac{(x_i-\bar{X})(y_i-\bar{Y})}{n}$$

If $cov(X,Y) > 0$, then X and Y tend move together.
If $cov(X,Y) < 0$ then X and Y have an opposite effect on eachother.
And $cov(X,Y) = 0$?

COVARIANCE

Covariance: measure of how two variables change together.

Two series X and Y of values, each of size n.

$$cov(X,Y) = \overline{(X-\bar{X})(Y-\bar{Y})} = \sum_{i=1}^{n} \frac{(x_i-\bar{X})(y_i-\bar{Y})}{n}$$

Example:

$$\begin{aligned}
X &= [4,2,3,4,5,5,3,1,1,2]\\
Y &= [6,4,3,5,7,8,5,3,3,2]\\
\bar{X} &= 3\\
\bar{Y} &= 4.6\\
X-\bar{X} &= [1,-1,0,1,2,2,0,-2,-2,-1]\\
Y-\bar{Y} &= [1.4,-0.6,-1.6,0.4,2.4,3.4,0.4,-1.6,-1.6,-2.6]\\
(X-\bar{X})(Y-\bar{Y}) &= [-0.84,1.56,2.56,-0.24,0.96,1.36,-0.64,5.76,5.76,6.76]\\
cov(X,Y) &= 2.3
\end{aligned}$$

COVARIANCE

A covariance matrix is a matrix showing the covariance of two or more variables to each other.

- If one variable changes, does the other variable change as well and in what direction?
- Example: altitude, temperature and air pressure

altitude	T °C	pressure
0	20	1
1000	10	0.9
2000	0	0.8
3000	-10	0.7
4000	-20	0.5
5000	-30	0.3

COVARIANCE

A covariance matrix is a matrix showing the covariance of two or more variables to each other.

- If one variable changes, does the other variable change as well and in what direction?
- Example: altitude, temperature and air pressure

altitude	T °C	pressure
0	20	1
1000	10	0.9
2000	0	0.8
3000	-10	0.7
4000	-20	0.5
5000	-30	0.3

	alt.	**T**	**P**
alt.	2916667	-29166.7	-400
T	-29166.7	291.667	4
P	-400	4	0.057

KALMAN FILTER

THE BASICS

The Kalman filter needs a number of parameters to run.

These come from the **process equations**: equations that describe how the state of the system in the next time step depends on the current state and any changes that happen to the system.

THE BASICS

The Kalman filter needs a number of parameters to run.

These come from the **process equations**: equations that describe how the state of the system in the next time step depends on the current state and any changes that happen to the system.

For example: a car drives down the road. Its position at time $t+1$ depends on its position at time t, the control input at t (is the car braking or accelerating) and system dynamics (it slows down due to friction).

THE BASICS

The Kalman filter needs a number of parameters to run.

These come from the **process equations**: equations that describe how the state of the system in the next time step depends on the current state and any changes that happen to the system.

For example: a car drives down the road. Its position at time $t + 1$ depends on its position at time t, the control input at t (is the car braking or accelerating) and system dynamics (it slows down due to friction).

Instead of t, we use k to denote *discrete time steps*.

THE PROCESS EQUATIONS (1)

The process is governed by a linear difference equation:

The state at time step *k*

The state one time step ago

$$x_k = F \cdot x_{k-1} + B \cdot u_{k-1} + w_{k-1}$$

THE PROCESS EQUATIONS (1)

The process is governed by a linear difference equation:

The state at time step *k*

The state one time step ago

$$x_k = F \cdot x_{k-1} + B \cdot u_{k-1} + w_{k-1}$$

$n \times n$ matrix changing the previous state into the current state

THE PROCESS EQUATIONS (1)

The process is governed by a linear difference equation:

The state at time step k

The state one time step ago

Control input, a vector of size l

$$x_k = F \cdot x_{k-1} + B \cdot u_{k-1} + w_{k-1}$$

$n \times n$ matrix changing the previous state into the current state

$n \times l$ matrix that maps the control input onto the state

THE PROCESS EQUATIONS (1)

The process is governed by a linear difference equation:

$$x_k = F \cdot x_{k-1} + B \cdot u_{k-1} + w_{k-1}$$

The state at time step *k*

The state one time step ago

Control input, a vector of size *l*

n × *n* matrix changing the previous state into the current state

n × *l* matrix that maps the control input onto the state

The process noise, a vector of size *n*.

THE PROCESS EQUATIONS (2)

We take m measurements which will be related to the state x according to:

$$z_k = H \cdot x_k + v_k$$

Measurements, a vector of size m

$m \times n$ matrix mapping the state to the measurements

Measurement noise, a vector of size m

THE PROCESS EQUATIONS: RECAP OF MAIN MODELS

- **State transition model** F: matrix $n \times n$ that describes how the state changes from $k-1$ to k without controls or noise.
- **Control input model** B: matrix $n \times l$ that describes how the control u_{k-1} changes the state from $k-1$ to k.
- **Observation model** H: Matrix $m \times n$ that describes how to map the state x_k to the measurements z_k.

THE PROCESS EQUATIONS: RECAP OF MAIN MODELS

- **State transition model** F: matrix $n \times n$ that describes how the state changes from $k-1$ to k without controls or noise.
- **Control input model** B: matrix $n \times l$ that describes how the control u_{k-1} changes the state from $k-1$ to k.
- **Observation model** H: Matrix $m \times n$ that describes how to map the state x_k to the measurements z_k.

- **Process noise model** w_k: a vector of size n
- **Measurement noise model** v_k: a vector of size m

THE PROCESS EQUATIONS (3)

The variables w_k and v_k contain the random noise on the state and measurements. They (are assumed to) have a **normal** distribution.

$$p(w) \sim \mathcal{N}(0, Q)$$
$$p(v) \sim \mathcal{N}(0, R)$$

With a **covariance matrix** of Q and R; this reflects the width of the normal distribution

p means probability distribution

$\mathcal{N}$ is the notation for a normal distribution

ROUND AND ROUND GOES THE KALMAN FILTER

The goal of a Kalman filter is to **estimate** the state *x* at each time step given

- noisy measurements,
- control input,
- the process equations.

The state of the filter is represented by two variables:

- $\hat{\mathbf{x}}_{k|k}$: the state estimate at time k given observations up to and including at time k
- $\mathbf{P}_{k|k}$: the ***error covariance matrix*** (a measure of the estimated accuracy of the state estimate)

ROUND AND ROUND GOES THE KALMAN FILTER

The goal of a Kalman filter is to **estimate** the state x at each time step given

- noisy measurements,
- control input,
- the process equations.

the notation $\hat{\mathbf{x}}_{n|m}$ represents the **estimate** of $\mathbf{x}$ at time n given observations up to and including at time $m \leq n$

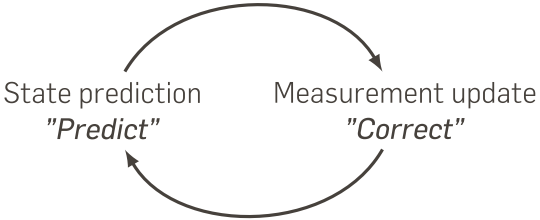
The state of the filter is represented by two variables:

- $\hat{\mathbf{x}}_{k|k}$: the state estimate at time k given observations up to and including at time k
- $\mathbf{P}_{k|k}$: the *error covariance matrix* (a measure of the estimated accuracy of the state estimate)

ROUND AND ROUND GOES THE KALMAN FILTER

The Kalman filter continuously loops through two steps

- The **state prediction** step.
- The **measurement update** step.



ROUND AND ROUND GOES THE KALMAN FILTER

The **Prediction** step uses the state estimate from the previous timestep to produce an estimate of the state **at the current timestep**. This is called the **a priori** estimate

- *A priori* means that the estimate is taken before any new sensor measurements have come in.
- **Notation**: $\hat{\mathbf{x}}_{k|k-1}$

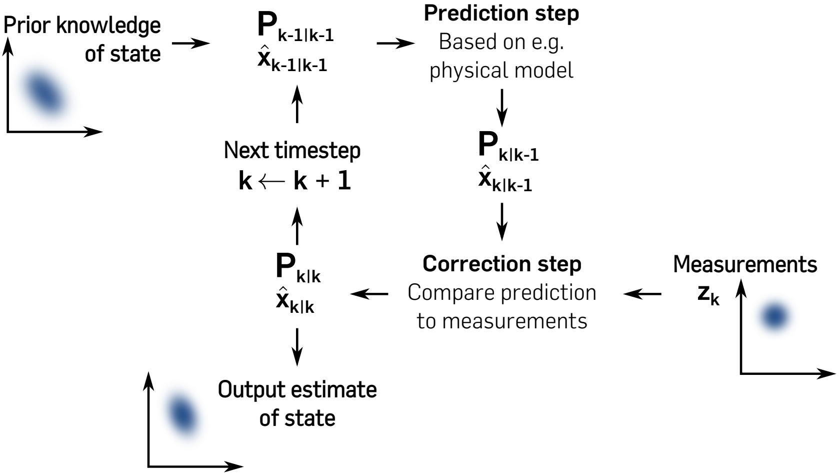
ROUND AND ROUND GOES THE KALMAN FILTER

The **Prediction** step uses the state estimate from the previous timestep to produce an estimate of the state **at the current timestep**. This is called the **a priori** estimate

- *A priori* means that the estimate is taken before any new sensor measurements have come in.
- **Notation**: $\hat{\mathbf{x}}_{k|k-1}$

The **Correction** step is run after new measurements have come in an provide the **a posteriori** estimate.

- *A posteriori* because the estimate is made after new sensor measurements have come in.
- **Notation**: $\hat{\mathbf{x}}_{k|k}$



Prior knowledge of state
$\mathbf{P}_{k-1|k-1}$
$\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step
Based on e.g. physical model
$\mathbf{P}_{k|k-1}$
$\hat{\mathbf{x}}_{k|k-1}$
Next timestep
$\mathbf{k \leftarrow k + 1}$
$\mathbf{P}_{k|k}$
$\hat{\mathbf{x}}_{k|k}$
Correction step
Compare prediction to measurements
Measurements
$\mathbf{z}_k$
Output estimate of state

PREDICT STEP EQUATIONS

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F} \cdot \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B} \cdot u_{k-1}$$

A priori estimate of the state at time k

A posteriori estimate of the state at time $k-1$

$$\mathbf{P}_{k|k-1} = \mathbf{F} \cdot \mathbf{P}_{k-1|k-1} \cdot \mathbf{F}^{\mathsf{T}} + \mathbf{Q}$$

A priori estimate of the error covariance at time k

A posteriori estimate of the error covariance at time $k-1$

Process noise

CORRECT STEP EQUATIONS (MEASUREMENT UPDATE)

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \cdot \mathbf{H}^{\mathsf{T}} \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^{\mathsf{T}} + \mathbf{R})$$

The **Kalman gain**, this needs to calculated first

Sensor noise

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1})$$

The ***a posteriori*** estimated state

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1}$$

The ***a posteriori*** estimated covariance of our state

CORRECT STEP EQUATIONS (MEASUREMENT UPDATE)

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top + \mathbf{R})$$

estimated state at last step

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1})$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1}$$

CORRECT STEP EQUATIONS (MEASUREMENT UPDATE)

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \cdot \mathbf{H}^T \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^T + \mathbf{R})$$

(actual measurements - expected measurement) $\times$ Kalman gain

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \boxed{\mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1})}$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1}$$

THE KALMAN FILTER LOOP

Prediction

1. Project the state ahead
   $\hat{\mathbf{x}}_{k|k-1} = \mathbf{F} \cdot \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B} \cdot u_{k-1}$
2. Project the error covariance ahead
   $\mathbf{P}_{k|k-1} = \mathbf{F} \cdot \mathbf{P}_{k-1|k-1} \cdot \mathbf{F}^{\mathsf{T}} + \mathbf{Q}$

Correction

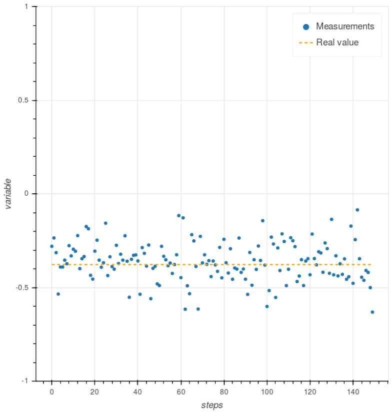
1. Compute Kalman gain
   $\mathbf{K}_k = \mathbf{P}_{k|k-1} \cdot \mathbf{H}^{\mathsf{T}} \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^{\mathsf{T}} + \mathbf{R})^{-1}$
2. Update estimate with measurements $\mathbf{z}_k$
   $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1})$
3. Update the error covariance
   $\mathbf{P}_{k|k} = (I - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1}$

Initial estimates
for $\hat{\mathbf{x}}_{k-1}$ and $\mathbf{P}_{k-1}$

EXAMPLES

EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

A Kalman filter to estimate the state of a system with **one variable**. For this demonstration, the variable remains **constant** (for example, measuring a voltage or a temperature).



EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

Prediction:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F} \cdot \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B} \cdot u_{k-1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F} \cdot \mathbf{P}_{k-1|k-1} \cdot \mathbf{F}^{\mathsf{T}} + \mathbf{Q}$$

EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

Prediction:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F} \cdot \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B} \cdot u_{k-1}$$

$\mathbf{x} = \begin{bmatrix} x \end{bmatrix}$

$\mathbf{F} = \begin{bmatrix} 1 \end{bmatrix}$

$u = \begin{bmatrix} 0 \end{bmatrix}$

$$\mathbf{P}_{k|k-1} = \mathbf{F} \cdot \mathbf{P}_{k-1|k-1} \cdot \mathbf{F}^{\top} + \mathbf{Q}$$

EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

Prediction:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F} \cdot \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B} \cdot u_{k-1} = \hat{\mathbf{x}}_{k-1|k-1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F} \cdot \mathbf{P}_{k-1|k-1} \cdot \mathbf{F}^{\mathsf{T}} + \mathbf{Q}$$

EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

Prediction:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F} \cdot \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B} \cdot u_{k-1} = \hat{\mathbf{x}}_{k-1|k-1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F} \cdot \mathbf{P}_{k-1|k-1} \cdot \mathbf{F}^{\mathsf{T}} + \mathbf{Q} = \mathbf{P}_{k-1|k-1}$$

EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

Correction:

observation matrix $\mathbf{H} = \begin{bmatrix}1\end{bmatrix}$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top + \mathbf{R})^{-1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1})$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1}$$

EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

Correction:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top + \mathbf{R})^{-1} \\ &= \mathbf{P}_{k|k-1} \cdot (\mathbf{P}_{k|k-1} + \mathbf{R})^{-1}\end{aligned}$$

R?

$\mathbf{R} = \begin{bmatrix} 0.01 \end{bmatrix}$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1})$$

$$\mathbf{P}_{k|k} = (\mathcal{I} - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1}$$

EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

Correction:

$$
\begin{aligned}
\mathbf{K}_k &= \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top + \mathbf{R})^{-1} \\
&= \mathbf{P}_{k|k-1} \cdot (\mathbf{P}_{k|k-1} + \mathbf{R})^{-1} \\
&= \mathbf{P}_{k|k-1} \cdot (\mathbf{P}_{k|k-1} + [0.01])^{-1}
\end{aligned}
$$

$$
\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1})
$$

$$
\mathbf{P}_{k|k} = (I - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1}
$$

EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

Correction:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\mathsf{T} \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\mathsf{T} + \mathbf{R})^{-1} \\ &= \mathbf{P}_{k|k-1} \cdot (\mathbf{P}_{k|k-1} + \mathbf{R})^{-1} \\ &= \mathbf{P}_{k|k-1} \cdot (\mathbf{P}_{k|k-1} + [0.01])^{-1}\end{aligned}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1}) = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \hat{\mathbf{x}}_{k|k-1})$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1}$$

EXAMPLE 1: MEASURING A NOISY YET CONSTANT VALUE

Correction:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^\top + \mathbf{R})^{-1} \\ &= \mathbf{P}_{k|k-1} \cdot (\mathbf{P}_{k|k-1} + \mathbf{R})^{-1} \\ &= \mathbf{P}_{k|k-1} \cdot (\mathbf{P}_{k|k-1} + [0.01])^{-1}\end{aligned}$$

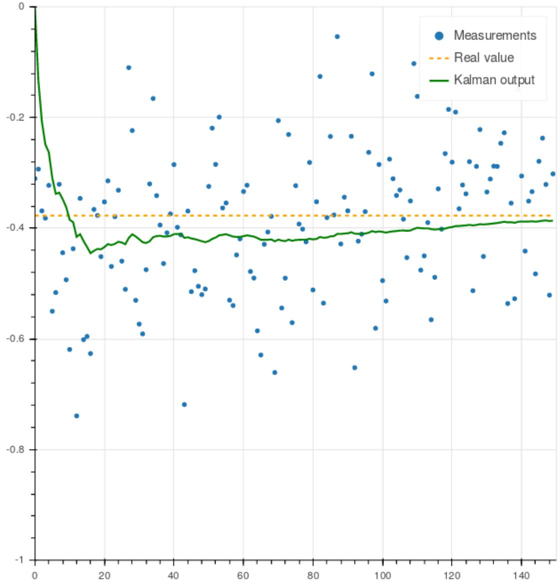
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1}) = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \hat{\mathbf{x}}_{k|k-1})$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1} = ([1] - \mathbf{K}_k) \cdot \mathbf{P}_{k|k-1}$$

IN PYTHON...

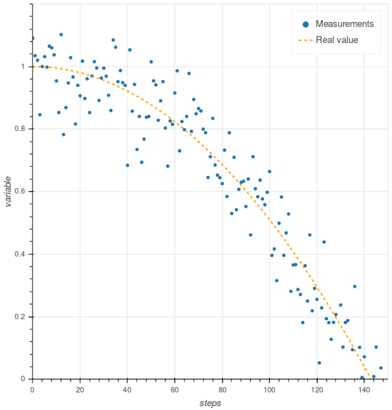
```python
from numpy.matlib import matrix

# initial state estimate
x = [matrix([0.])]
P = [matrix([0.001])]

k = 1 # step

R = matrix([0.01]) # estimate of measurement noise

def kalman(k):
    # Prediction phase
    x_prior = x[k-1]
    P_prior = P[k-1]

    # Correction phase
    K = P_prior * ( P_prior + R ).I
    x.append(x_prior + K * (z[k] - x_prior))
    P.append((matrix([1.]) - K) * P_prior)
```

The complete examples are online.


Measurements
Real value
Kalman output
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
140

EXAMPLE 2: FREE FALL

A Kalman filter to estimate the state of a system with **one variable** (height). The variable **changes** under the effect of an external force (gravity).



EXAMPLE 2: FREE FALL

Free fall equations

$$
\begin{aligned}
&\ddot{y}(t) = -g \\
\Rightarrow\ &\dot{y}(t) = \dot{y}(t_0) - g(t - t_0) \\
\Rightarrow\ &y(t) = y(t_0) + \dot{y}(t_0)(t - t_0) - \frac{g}{2}(t - t_0)^2
\end{aligned}
$$

EXAMPLE 2: FREE FALL

Free fall equations

$$
\begin{aligned}
&\ddot{y}(t) = -g \\
\Rightarrow\ &\dot{y}(t) = \dot{y}(t_0) - g(t - t_0) \\
\Rightarrow\ &y(t) = y(t_0) + \dot{y}(t_0)(t - t_0) - \frac{g}{2}(t - t_0)^2
\end{aligned}
$$

As a discrete time system, with time increment $t - t_0 = 1$:

$$y_k = y_{k-1} + \dot{y}_{k-1} - \frac{g}{2}$$

How to fit it into our process equations?

EXAMPLE 2: FREE FALL

The trick consists in embedding the velocity in our state: $\mathbf{x} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$

EXAMPLE 2: FREE FALL

The trick consists in embedding the velocity in our state: $\mathbf{x} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$

$$\begin{aligned}
\begin{bmatrix} y \\ \dot{y} \end{bmatrix}_{k|k-1} &= \mathbf{F} \cdot \begin{bmatrix} y \\ \dot{y} \end{bmatrix}_{k-1|k-1} + \mathbf{B} \cdot u_{k-1} \\
&= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} y \\ \dot{y} \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \cdot (-g)
\end{aligned}$$

EXAMPLE 2: FREE FALL

The trick consists in embedding the velocity in our state: $\mathbf{x} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$

$$
\begin{aligned}
\begin{bmatrix} y \\ \dot{y} \end{bmatrix}_{k|k-1} &= \mathbf{F} \cdot \begin{bmatrix} y \\ \dot{y} \end{bmatrix}_{k-1|k-1} + \mathbf{B} \cdot u_{k-1} \\
&= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} y \\ \dot{y} \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \cdot (-g)
\end{aligned}
$$

Measurements:

$$
\begin{aligned}
z_k &= \mathbf{H} \cdot \begin{bmatrix} y \\ \dot{y} \end{bmatrix}_{k|k} + \mathbf{w}_k \\
&= \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} y \\ \dot{y} \end{bmatrix}_{k|k} + \mathbf{w}_k
\end{aligned}
$$

EXAMPLE 2: FREE FALL

$$\Rightarrow \mathbf{F} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

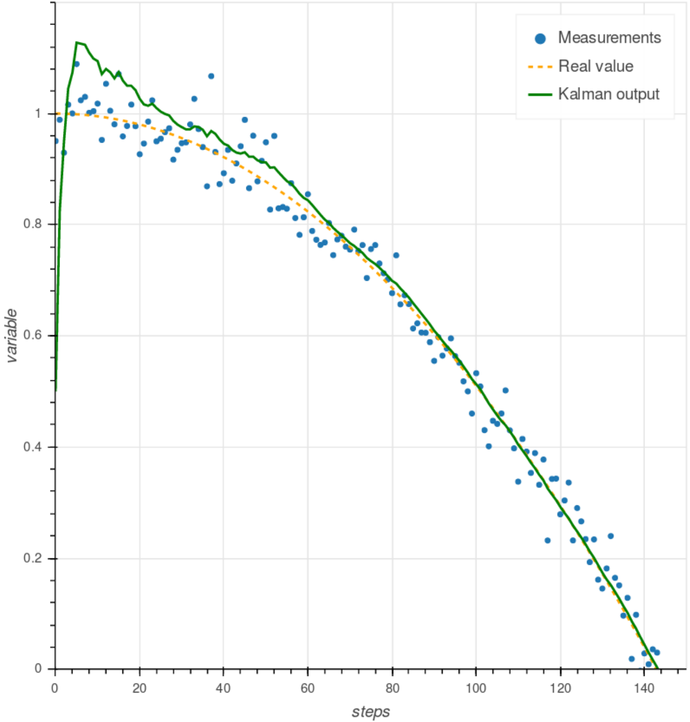
$$\mathbf{B} = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$u = \begin{bmatrix} -g \end{bmatrix}$$

IN PYTHON...

```
from numpy.matlib import matrix

x = [matrix([[0.5],[0.]])] # [y, dy]
P = [matrix([[0.001, 0.],[0., 0.001]])]
k = 1

F = matrix([[1.,1.],[0.,1.]]) # based on the free fall equations
B = matrix([[0.5],[1.]]) # the contribution of the gravity g
H = matrix([1.,0.]) # we only measure the height, not the velocity
Q = matrix([[0.],[0.]]) # no process noise
R = matrix([0.001]) # estimate of our measurement noise
u = matrix([-g]) # control input: gravity

def kalman(k):
    # Prediction phase
    x_prior = F * x[k-1] + B * u
    P_prior = F * P[k-1] * F.T + Q

    # Correction phase
    K = P_prior * H.T * ( H * P_prior * H.T + R ).I
    x.append(x_prior + K * (z[k] - H * x_prior))
    P.append((matrix([[1,0],[0,1]]) - K * H) * P_prior)
```



Measurements
Real value
Kalman output
variable
steps
0
0.2
0.4
0.6
0.8
1
20
40
60
80
100
120
140

FAQ

Is a Kalman Filter similar to ***complementary filters***?

- Complementary filters are often used to combine accelerometer and gyro readings on an IMU.
- CF is simple (few lines of code, no matrices) and combines a high and low pass filter.
- CF does not predict states into the future.

More about complementary filters

What if my problem is non-linear?

- There are alternative versions out there such as the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF).

FURTHER READING

- A good presentation on Kalman filter
- Lesson 2 of Artificial Intelligence for Robotics at Udacity

Video demonstrations

Kalman filter on accelerometer and gyro to read stable angle

- **http://www.youtube.com/watch?v=MJ71V_wxtuU**
- **http://www.youtube.com/watch?v=Y3TzhXYF0Lg**

Kalman filter tracking an airplane

- **http://www.youtube.com/watch?v=0GSIKwfkFCA**

That's all, folks!

Questions:
Portland Square A216 or **severin.lemaignan@plymouth.ac.uk**

Slides:
github.com/severin-lemaignan/module-mobile-and-humanoid-robots