Kalman Filters and Sensor Fusion

Girish Chowdhary

Associate Professor,
Director Distributed Autonomous Systems Lab
University of Illinois at Urbana Champaign,
www.daslab.illinois.edu

April 29, 2021

Outline

- Recall the basics of probability theory
- Understand, PDF, CDF, Mean, and Variance
- Recall the Gaussian probability distribution and the Gaussian noise model

Axioms of Probability

- $p(A)$ denotes the probability that the event A is true: Axioms of prob
  - $p(A) \geq 0$ (Probability is a positive number assigned to the event)
  - $P(C) = 1$ The probability of the certain event is 1
  - If A and B are mutually exclusive, then $p(A + B) = p(A) + p(B)$

Random variable

- (discrete) Random variable (RV): a variable that can take one of many values
- Denote the probability of event $X = x$ by $p(X = x)$ or simply $p(x)$
- **here $p(x)$ is the probability mass function**
  - $0 \geq p(x) \leq 1$ (The probability of some event happening between zero and one)
  - $\sum_{x \in X} p(x) = 1$ (Something happens)

Frequency definition

- $p(A) = \lim_{n \to \infty} \frac{n_A}{n}$, where n_A is the number of occurrences of A and n is the number of trials
- Classical definition $\Rightarrow$ For the random variable X $p(X = x_i) = c_i/N$ where N is the number of possible outcomes, c_i is the number of outcomes favorable to $X = x_i$
- e.g. even die roll: $\frac{3}{6}$
- However, the classical definition gives weired results, so the frequency definition is preferred

Rules

- For X be able to take any value x_i and RV Y be able to take any value y_i
- If c_i is the number of trials in which $X = x_i$ over a set of N trials, then frequentist definition of probability: $p(X = x_i) = c_i/N$ as $N \to \infty$
- Question: what is the joint probability that I will get "snake eyes"? i.e.$x_i = 1, y_i = 1$
- Joint probability: in the game of Craps, let X be the number of dots on a side of a dice, and Y on another

$$p(X = x_i, Y = y_i) = \frac{n_{ij}}{N}$$

- Here n_{ij} is the number of trials over which $X = x_i, Y = y_i$

Rules of probability

- Probability of union of two events A, B, i.e. probability of **A or B**

$$\begin{aligned} p(A \vee B) &= p(A) + p(B) - P(A \wedge B) \\ &= p(A) + p(B) \quad \text{If A and B are mutually exclusive} \end{aligned}$$

- Joint event $p(A, B)$ **Product Rule**

$$p(A, B) = p(A \wedge B) = p(A|B)p(B)$$

- Given $p(A, B)$ we define the marginal distribution over B
- Called as the Sum rule or rule of total probability

$$p(A) = \sum_b p(A|B) = \sum_b p(A|B = b)p(B = b)$$

Basic rules of probability

If we are only concerned about the probability of one variable, we can *marginalize* or sum over the other variable, this leads to $p(X = x_i) = \sum_{j=1}^{L} p(X = x_i, Y = y_j)$. This leads to the sum rule

Sum rule

$$p(X) = \sum_{Y} p(X, Y) = \sum_{Y} p(X|Y = y)p(Y = y)$$

Conditional probability $p(Y = y_j|X = x_i)$

Product rule

$$p(X, Y) = p(Y|X)p(X)$$

Conditional Probability

Manipulating the Product Rule

$$p(X|Y) = \frac{p(X,Y)}{p(Y)}$$

Bayes Theorem

$$p(X|Y) = \frac{p(X,Y)}{p(Y)} = \frac{p(Y|X)p(X)}{p(Y)}$$

The denominator can be expressed as the *Total Probability* $p(Y) = \sum_{x'} p(Y = y|X = x')p(X = x')$. This is a normalization constant required to ensured that the lhs of Bayes rule over all values of Y is equal to 1 To get here, we never needed the frequency definition of probability

Example Bayes rule

Example 2.2.3.1 from Murphy

- Test sensitivity 80%, i.e. when you have cancer (y=1), test will be true (x=1): $p(x = 1|y = 1) = 0.8$
- This is a case of a high likelihood of measuring x when the state is y
- But what is the *prior* probability of being in state y?: $\Rightarrow p(y) = 0.004$
- Clearly, now $p(cancer = 1|test = 1) = p(y = 1|x = 1) \propto p(x|y)p(y) = 0.8 \times 0.004$
- But we must account for the total probability $p(x)$, which includes false positive $p(x = 1|y = 0) = 0.1$

So Bayes law tells us:

$$p(y = 1|x = 1) = \frac{p(x = 1|y = 1)p(y = 1)}{p(x = 1|y = 1)p(y = 1) + p(x = 1|y = 0)p(y = 0)}$$
$$= 0.031$$

Probability Density Functions (PDFs)

Concept of probability can be extended to continuous variables using the PDF

PDF

$$p(x \in (a, b)) = \int_a^b p(x)dx$$

- PDFs satisfy the rules of probability:

$$p(x) \geq 0$$

$$\int_{-\infty}^{\infty} p(x)dx = 1$$

- Sum and Product rules apply to pdfs:

$$p(x) = \int p(x, y)dy, \quad p(x, y) = p(y|x)p(x)$$

Expectations

Expectation: Average value of a function

Expectation of a continuous pdf

$$\mathbb{E}[f] = \int p(x)f(x)dx$$

Notice LHS does not have x in it, why?

Expectation of function of several variables can be taken wrt a variable, e.g. for $f(x, y)$

$$\mathbb{E}_x[f] = \int \int p(x, y)f(x, y)dydx$$

$\mathbb{E}_x[f]$ is a function of y

Conditional expectation

$$\mathbb{E}_x[f(x|y)] = \int p(x|y)f(x)dx$$

Variance

Variance known as the second moment

Variance

$$Var[f] = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2]$$

Useful identity

Variance

$$Var[f] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$$

The Gaussian distribution:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{1}{2\pi\sigma^2}(x-\mu)^2\}$$

hyperparameters: Mean: μ, variance σ^2, std deviation σ Figuring out Gaussian distribution hyperparameters

Gaussian Distribution

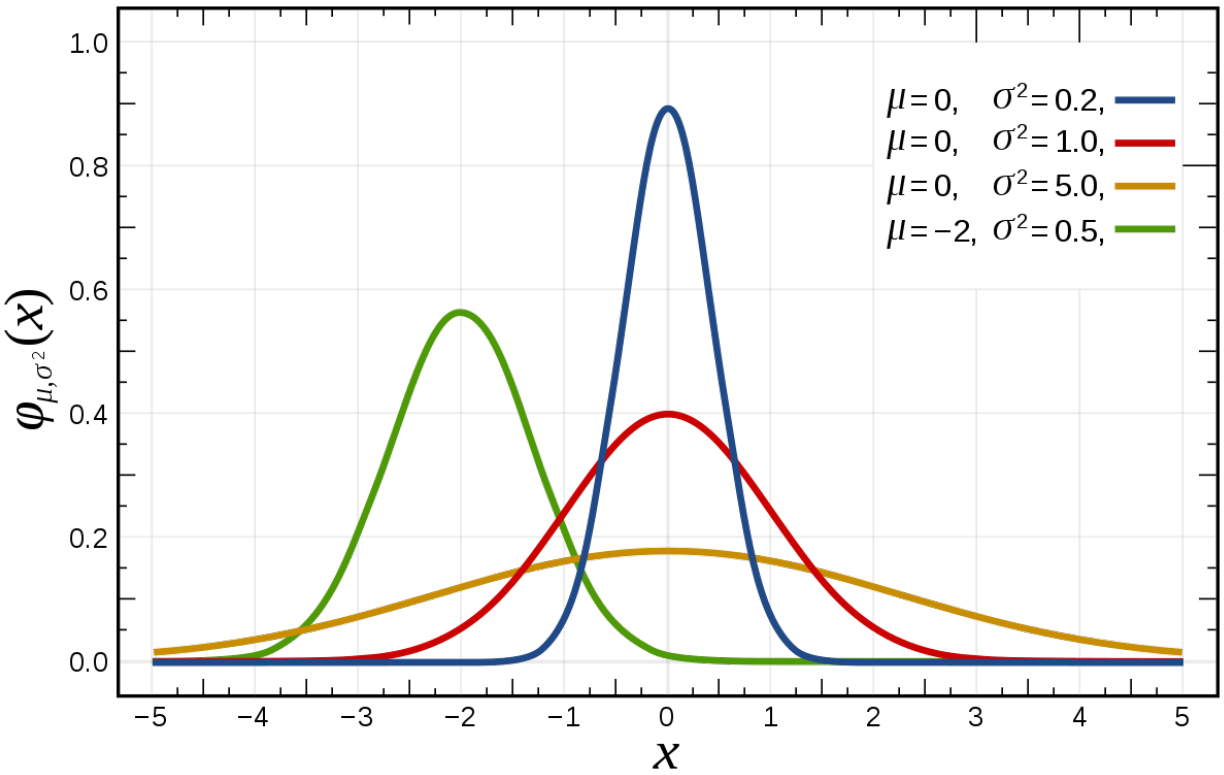


Figure: Gaussian Distribution

image source

Gaussian Distribution

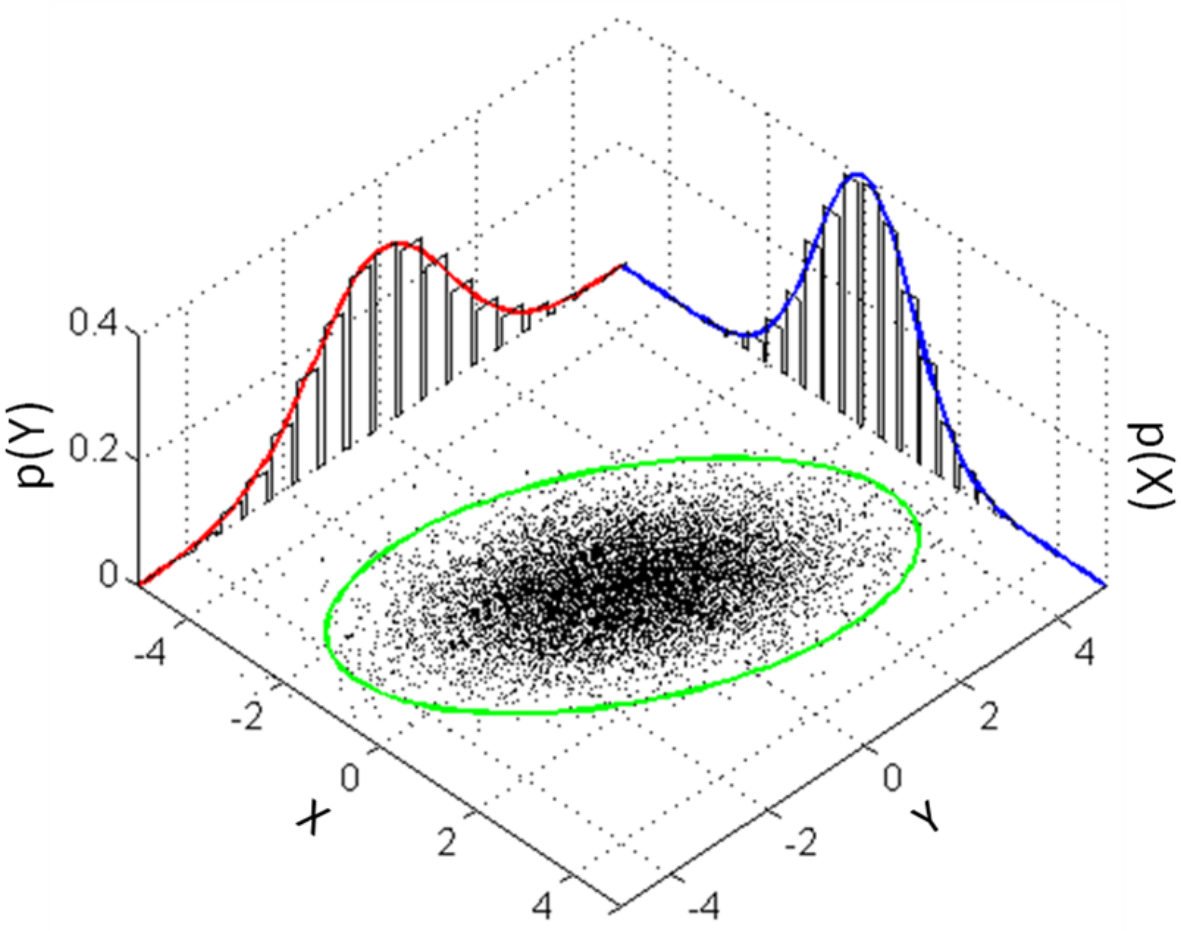


Figure: Samples from Multi-variate Gaussian Distribution

https://upload.wikimedia.org/wikipedia/commons/8/8e/MultivariateNormal.png

Goals

- State Space representation
- Definitions of state, state variables, state vector, state space, state trajectory, input and output
- State space representation of dynamic systems
- Solution of state space models
- Observer design
- Observability

Definitions

- State: The state of a dynamic system is the smallest set of variables (called state variables) such that the knowledge of these variables at initial time $t = t_0$, together with the knowledge of the input for $t \geq t_0$, completely determines the behavior of the system for $t \geq t_0$.

State

Current state, control system, and system model completely defines future response

- State Variables: The variables that make up the state of the dynamic system. If n variables, $(x_1, x_2, \ldots, x_n)$, are required to completely describe the behavior of the system, then such n variables are a set of state variables.

State Vector

- State Vector: The n-dimensional column vector, x, composed of the state variables: $x \in \mathbb{R}^n$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- State Space: The n-dimensional space whose coordinates axes consists of the x_1-axis, x_2-axis, ..., x_n-axis, where $x_1, x_2, \ldots, x_n$ are the state variables.
  For Example: If a system has two variables, i.e., $n = 2$, then the state-space is a two-dimensional space with x_1 as the abscissa and x_2 as the ordinate.

State-Space modeling

- State Trajectory: The path representing the state of the system in a state space: x
- Inputs: Inputs describe the external excitation of the dynamics; and are extrinsic to the system dynamics: $u(t)$
- Outputs: Outputs describe the directly measured variables; outputs are a function of the state variables and inputs; outputs are not independent variables: $y(t)$

General form of state space models

Assume that a system has n state variables, m inputs and r outputs. Then the general form of the state space model for the system is

$$\begin{aligned}
\dot{x}_1(t) &= f_1(x_1, x_2, \ldots, x_n; u_1, u_2, \ldots, u_m; t) \\
\dot{x}_2(t) &= f_2(x_1, x_2, \ldots, x_n; u_1, u_2, \ldots, u_m; t) \\
&\vdots \\
\dot{x}_n(t) &= f_n(x_1, x_2, \ldots, x_n; u_1, u_2, \ldots, u_m; t)
\end{aligned}$$

The above first-order differential equations in the state variables are called state differential equations or state space equations.
They can be nonlinear

State space models

The outputs $y_1(t), y_2(t), \ldots, y_r(t)$ are given by

$$\begin{aligned} y_1(t) &= g_1(x_1, x_2, \ldots, x_n; u_1, u_2, \ldots, u_m; t) \\ y_2(t) &= g_2(x_1, x_2, \ldots, x_n; u_1, u_2, \ldots, u_m; t) \\ &\vdots \\ y_r(t) &= g_r(x_1, x_2, \ldots, x_n; u_1, u_2, \ldots, u_m; t) \end{aligned}$$

• In compact vector form, we can write the state differential equations and outputs as

$$\begin{aligned} \dot{x}(t) &= f(x, u, t) \\ y(t) &= g(x, u, t) \end{aligned}$$

where $x(t)$ is the state vector, $u(t)$ is the input vector, and $y(t)$ is the output vector.

Linear Time Invariant Systems

If the system dynamics is linear and time-invariant, then the state space model is of the form

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1}$$
$$y(t) = Cx(t) + Du(t) \tag{2}$$

$x \in \mathbb{R}^{n\times 1}$ state vector
$u \in \mathbb{R}^{m\times 1}$ input vector
$y \in \mathbb{R}^{r\times 1}$ output vector

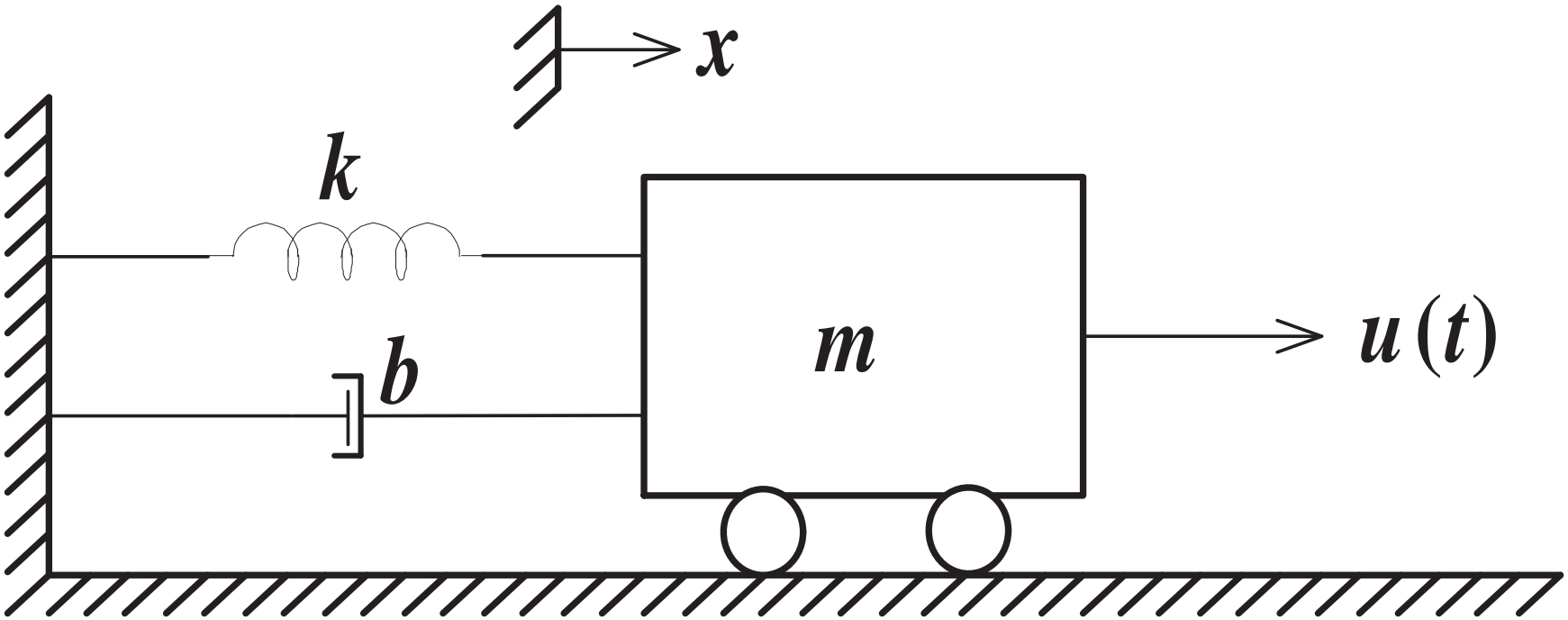
$A \in \mathbb{R}^{n\times n}$ state matrix
$B \in \mathbb{R}^{n\times m}$ input matrix
$C \in \mathbb{R}^{r\times n}$ output matrix
$D \in \mathbb{R}^{m\times m}$ direct transmission matrix

• The matrices A, B, C, D are called the system matrices and they depend on the physical parameters of the system such as mass, spring and damping coefficients etc

Comments

- State models are not unique: One can choose many different sets of state variables to describe the system in state space form
- Commonly used variables include:
  - variables associated with energy storage elements;
  - sets including one variable and its successive derivatives;
  - variables that give a canonical (standard) form for the state space representation

Ex: Mass Spring Damper Sys



• The state space model will depend on the set of state variables that are chosen.

Energy Variables

• The variables associated with energy storage elements are the force in the spring $F_k(t)$ and the velocity of the mass $v(t)$.

Elemental equations

$$F_b(t) = b\dot{x}(t) \tag{3}$$

$$\frac{dF_k(t)}{dt} = kv(t) \quad (F_k(t) = kx(t)) \tag{4}$$

$$m\frac{dv(t)}{dt} = u(t) - F_b(t) - F_k(t) \tag{5}$$

Energy Variables contd

Choose state variables as

$$x_1 = F_k$$
$$x_2 = v$$

State variable equations:

$$\dot{x}_1 = kx_2 \tag{6}$$
$$\dot{x}_2 = -\frac{1}{m}x_1 - \frac{b}{m}x_2 + \frac{1}{m}u \tag{7}$$

Output equation: Assuming that the displacement of the mass is measured, the output equation is

$$y = \frac{1}{k}x_1 \tag{8}$$

Energy Variables State Space model

The matrix form of the state space model is (with variables associated with energy storage elements as state variables)

$$\underbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}0 & k\\ -\frac{1}{m} & -\frac{b}{m}\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x} + \underbrace{\begin{bmatrix}0\\ \frac{1}{m}\end{bmatrix}}_{B} u \tag{9}$$

$$y = \underbrace{\begin{bmatrix}\frac{1}{k} & 0\end{bmatrix}}_{C}\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x} + \underbrace{[0]}_{D} u \tag{10}$$

Another set of state variables

Choose state variables as

$$x_1 = \text{displacement of the mass},$$
$$x_2 = \text{velocity of the mass},$$

and assume the measured output is the displacement of the mass. Then the state variable equations and output equation are

$$\dot{x}_1 = x_2 \tag{11}$$
$$\dot{x}_2 = -\frac{k}{m}x_1 - \frac{b}{m}x_2 + \frac{1}{m}u \tag{12}$$
$$y = x_1 \tag{13}$$

Another set contd

The matrix form of the state space model is (with displacement and velocity of the mass as state variables)

$$\underbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}0 & 1\\ -\frac{k}{m} & -\frac{b}{m}\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x} + \underbrace{\begin{bmatrix}0\\ \frac{1}{m}\end{bmatrix}}_{B} u \tag{14}$$

$$y = \underbrace{\begin{bmatrix}1 & 0\end{bmatrix}}_{C}\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x} + \underbrace{[0]}_{D} u \tag{15}$$

Solution of LTI systems

- Consider the first-order system described by the state equation

$$\dot{x}(t) = ax + bu, \; x(0) = x_0. \tag{16}$$

- The solution is given by

$$x(t) = e^{at}x_0 + \int_0^t e^{a(t-\tau)}bu(\tau)d\tau. \tag{17}$$

- Notice that the solution consists of two parts: the first part is due to the initial condition x_0, and the second part is due to the external input $u(t)$.

Vector case

- Consider the n-th order system given by

$$\dot{x}(t) = Ax(t) + Bu(t), \; x(0) = x_0 \tag{18}$$

where $x(t)$ is the n-dimensional state vector.
- The solution is given by

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau. \tag{19}$$

- In the case of higher-order systems ($n > 1$), e^{At} is a matrix, and is called the State Transition Matrix.
- The state transition matrix is denoted by $\Phi(t)$, that is,

$$\Phi(t) := e^{At} \tag{20}$$

Solution to LTI systems

- The solution vector can be written as

$$x(t) = \Phi(t)x_0 + \int_0^t \Phi(t-\tau)Bu(\tau)d\tau \tag{21}$$

- How does one compute $\Phi(t)$?
- Consider the infinite series expansion of e^{at}, that is,

$$e^{at} = 1 + at + \frac{(at)^2}{2!} + \frac{(at)^3}{3!} + \cdots \tag{22}$$

- For the matrix case we have

$$e^{At} = I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \cdots \tag{23}$$

where I denotes the $n \times n$ identity matrix.

Stability in dynamical systems

The problem of stability in dynamical systems is very simple and very complex. In the context of linear time invariant systems of the form

$$\dot{x} = Ax + Bu$$

The full and final condition for stability is:

The LTI system $\dot{x} = Ax + Bu$ is exponentially asymptotically stable if and only if the real part of the eigenvalues of A are negative, that is $\text{Real}(\lambda_i(A)) < 0 \ \forall \ i \ \in \text{spectrum}(A)$, or in other words, A is *Hurwitz*.

The state feedback idea

$$\dot{x} = Ax + Bu$$

Consider the full state feedback case: $C = I$
then the state feedback controller always has the form:

$$u = -Kx$$

where K is the gain matrix, and x is the state
Substituting this value of u, we have that

$$\dot{x} = (A - BK)x$$

So the stability of the system depends on the eigenvalues of $A - BK$

The estimation problem

- Imagine that you want to estimate the state of a system $\dot{x} = Ax + Bu$ from the measurements $y = Cx$, how do we do this?

The estimation problem

- Imagine that you want to estimate the state of a system $\dot{x} = Ax + Bu$ from the measurements $y = Cx$, how do we do this?
- We can do this by designing an **observer system**, which is another dynamical system that over time will track our dynamic system

The estimation problem

- Imagine that you want to estimate the state of a system $\dot{x} = Ax + Bu$ from the measurements $y = Cx$, how do we do this?
- We can do this by designing an **observer system**, which is another dynamical system that over time will track our dynamic system
- We assume that we know A, B, C, and define the state of the observer system to be $\hat{x}$

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x})$$

The estimation problem

- Imagine that you want to estimate the state of a system $\dot{x} = Ax + Bu$ from the measurements $y = Cx$, how do we do this?
- We can do this by designing an **observer system**, which is another dynamical system that over time will track our dynamic system
- We assume that we know A, B, C, and define the state of the observer system to be $\hat{x}$

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x})$$

- We can now study the error dynamics: $e = x - \hat{x}$, and $\dot{e} = \dot{x} - \dot{\hat{x}}$

$$\begin{aligned}\dot{e} &= Ax + Bu - A\hat{x} - Bu - L(y - C\hat{x}) \\ &= A(x - \hat{x}) - LC(x - \hat{x}) \\ &= (A - LC)e\end{aligned}$$

- so the stability of the observer depends on the eigenvalues of $A - LC$

Observability

- When can we guarantee that we can find an l such that $A - LC$ is guaranteed Hurwitz?
- This is similar to asking the following question: Given a sequence of measurements $y_1, y_2, ..., y_t$ can I estimate $x_1, x_2, ...x_t$ and x_0.
- It turns out that the following condition, if satisfied, guarantees observability: Let W be the observability matrix:
- A system is observable if $\text{rank}(W) = n$

$$W = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{24}$$

Discrete time systems

In robotics, we need to deal with updates that happen on computers, which are digital devices, hence updates happen in discrete time.

Discrete time systems

$$x_{k+1} = \Phi_k x_k + \Gamma u_k \quad (25)$$

$$y_k = C x_k \quad (26)$$

- $\Phi_k = e^{A\Delta t}$ where Δt is the discretization interval
- $\Gamma = B\Delta t$

Now note that

$$x_1 = \Phi x_0 \tag{27}$$
$$x_2 = \Phi x_1 = \Phi^2 x_0 \tag{28}$$
$$\vdots \tag{29}$$
$$x_n = \Phi x_{n-1} = \Phi^n x_0 \tag{30}$$

Hence, the transition function $\bar{\Phi}_k = \Phi^k$

Intuition behind observability matrix

For the discrete system, $y_k = Cx_k$, so it follows that

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} C\ x_0 \\ C\Phi\ x_0 \\ C\Phi^2\ x_0 \\ \vdots \\ C\Phi^{n-1}\ x_0 \end{bmatrix} = \begin{bmatrix} C \\ C\Phi \\ C\Phi^2 \\ \vdots \\ C\Phi^{n-1} \end{bmatrix} x_0. \tag{31}$$

We can rewrite this as

$$Y = Wx_0$$

Where W is the observability matrix from (24). Now its clear that the above equation has a solution only if W has full row rank of n.

Outline

Class outline and outcomes

In this class we will learn about Kalman Filters, the learning outcomes are:

- Understand the Origin, need, and applications of Kalman Filters (KF)
- Learn about applications in robotics
- Get familiar with foundations of necessary probability theory
- Get familiar with the terminology, including process noise, measurement noise, predictive covariance, Gaussian white noise etc.
- Mathematically formulate the Kalman filtering problem
- Understand the KF algorithm and how to tune the filter
- Recognize when advanced filtering techniques are necessary and when KF assumptions are violated

What is Kalman filter

- **The filtering problem**: Find the best estimate of the true value of a system's state given noisy measurements of some values of that system's states
- What should a good filter do?
  - Provide an accurate and un-biased estimate
  - Provide confidence in its estimate

What is Kalman filter

- **The filtering problem**: Find the best estimate of the true value of a system's state given noisy measurements of some values of that system's states
- What should a good filter do?
  - Provide an accurate and un-biased estimate
  - Provide confidence in its estimate

The Kalman Filter

The Kalman filter is an optimal estimator for estimating the states of a linear dynamical system from sensor measurements corrupted with Gaussian white noise of some of that system's states.

What is Kalman filter

- **The filtering problem**: Find the best estimate of the true value of a system's state given noisy measurements of some values of that system's states
- What should a good filter do?
  - Provide an accurate and un-biased estimate
  - Provide confidence in its estimate

The Kalman Filter

The Kalman filter is an optimal estimator for estimating the states of a linear dynamical system from sensor measurements corrupted with Gaussian white noise of some of that system's states.

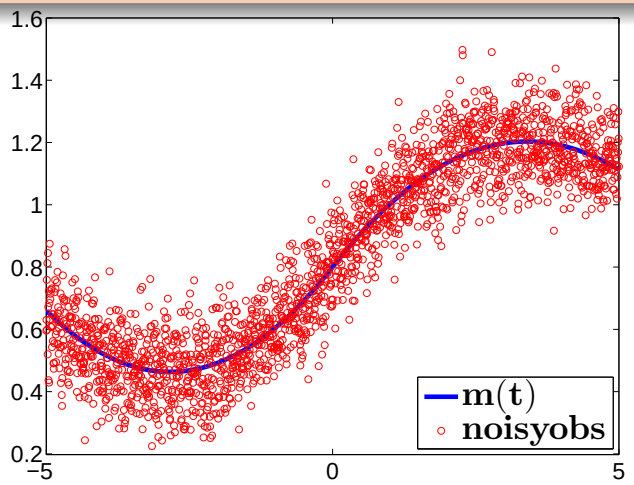


Figure: Noisy data and mean

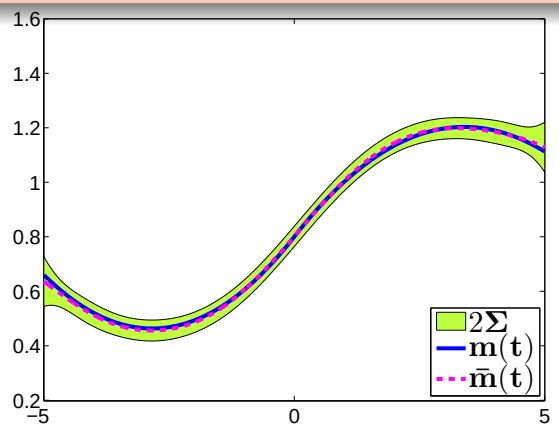


Figure: Estimate of the mean with predictive covariance

Why is the Kalman Filter so popular?

- It Works!

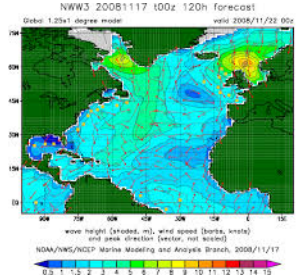
Why is the Kalman Filter so popular?

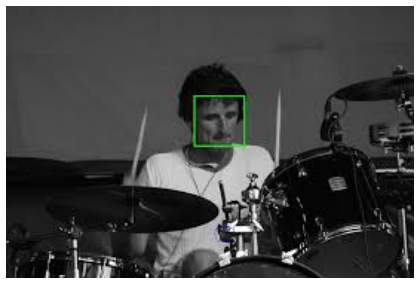
- It Works!
- Well... It works good-enough for many real-world applications
- Why?
  - Sensor noise does tend to be Gaussian in the limit of data received (central limit theorem)
  - Most systems behave linearly in local regions
  - Kalman filter utilizes feedback, which makes it robust to uncertainties
  - Its convenient to implement in an on-line manner to process streaming data
  - Amenable to real-time implementation for many problems

What is a Kalman Filter used for?

- The Kalman filter finds many many applications across pretty much all important scientific disciplines
- Its early application was on trajectory estimation on the Apollo space-craft
- Since then, it has been applied to many dynamical system state estimation problems, including: Cellphone, GPS, weather monitoring, precision agriculture, digital camera, ag-sensors.






Kalman filtering problem setup

- The Kalman filter will return a mean (average) of the quantity being estimated and provide a predictive variance on its estimate given:
  - The process and measurement noise variance is known
  - The dynamical system model is known
- The Kalman filter is guaranteed to be the optimal un-biased estimator for the following case:
  - The noise in the sensors in Gaussian
  - The dynamical system is linear
  - Sufficient number of states of the dynamical system are measured (the system is *observable*)
- If these assumptions are violated, the Kalman filter will be sub-optimal

Outline

Preliminaries: Continuous Time Invariant Systems

- If the system dynamics is Linear and Time-Invariant (LTI), then the state space model is of the form

Noisy continuous-time LTI systems

$$\dot{x} = Ax + B\omega_t \tag{32}$$
$$y = Hx + v_t \tag{33}$$

$x \in \mathbb{R}^{n\times 1}$ state vector
$\omega \in \mathbb{R}^{n\times 1}$ (additive) process noise
$y \in \mathbb{R}^{l\times 1}$ sensor measurements
$v \in \mathbb{R}^{l\times 1}$ (additive) measurement noise

$A \in \mathbb{R}^{n\times n}$ state matrix
$B \in \mathbb{R}^{n\times m}$ the input matrix (here we are using it for inputting noise)
$H \in \mathbb{R}^{l\times n}$ output matrix

Q and R matrices

The assumptions on the process and measurement noise are:

- Zero mean, uncorrelated, i.e. $\omega_k \sim \mathbb{N}(0, \sigma_\omega^2)$, $v_k \sim \mathbb{N}(0, \sigma_v^2)$
- $E[(\omega_t, \omega_s)] = Q\delta(t-s)$, $E[(v_t, v_s)] = R\delta(t-s)$, where δ is the dirac delta function, s.t. $\delta(t-s) = 1$ when $t = s$.
- no cross correlation between ω_k and v_k, i.e. $cov(\omega_k, v_k) = 0$ for all k

Herein lies the "official" definition of Process and Measurement noise. What it is saying is that you can set the diagonal term as σ_ω^2 and σ_v^2

- Practically, Q matrix represents the confidence in the process model, larger the Q matrix, the less confident we are
- Practically, R matrix represents the confidence in the measurements from the correcting sensors, higher the R matrix, the less confident in the measurements we are

Preliminaries: Discrete time Linear Systems

- If the system dynamics is Linear and Time-Invariant (LTI), then the state space model is of the form

Noisy discrete-time systems

$$x_{k+1} = \Phi_k x_k + \Gamma_k \omega_k \tag{34}$$

$$y_k = H_k x_k + v_k \tag{35}$$

$x \in \mathbb{R}^{n \times 1}$ state vector
$\omega \in \mathbb{R}^{n \times 1}$ (additive) process noise
$y \in \mathbb{R}^{l \times 1}$ sensor measurements
$v \in \mathbb{R}^{l \times 1}$ (additive) measurement noise

$\Phi_k \in \mathbb{R}^{n \times n}$ discretized state transition matrix
$\Gamma_k \in \mathbb{R}^{n \times m}$ Discretized input matrix
$H_k \in \mathbb{R}^{l \times n}$ output matrix

- The matrices Φ, H are called the system matrices and they depend on the physical parameters of the system such as mass, growth rate...
- note that these are the discrete versions of the equations: $\dot{x} = A(t)x + B(t)u; y = C(t)x$, in MATLAB the command is $c2d$
- In particular, $\Phi_k = e^{A\Delta t}$, $\Gamma_k = B(t)\Delta t$, $H_k = C(t)$, where Δt is the sampling time (dt)
- The process noise ω encodes our uncertainty in the knowledge of the dynamical evolution
- The measurement noise v encodes sensor measurement uncertainty

Relationship between Q and Q_d

Q_d is the discrete time version of Q, remember, $E[(\omega_t, \omega_s)] = Q_k \delta(t-s)$
TO get Q_d we integrate the dynamics in the continuous time case:

$$x(k+1) = \Phi_k x(k) + \int_{t_k}^{t_k+\Delta t} e^{A(t_{k+1}-\lambda)} B(\lambda)\omega(\lambda) d\lambda \tag{36}$$

So we have

$$\omega_k = \int_{t_k}^{t_k+\Delta t} e^{A(t_{k+1}-\lambda)} B(\lambda)\omega(\lambda) d\lambda \tag{37}$$

The exact solution is (assuming white noise process, and computing $Q_{d_k} = cov(\omega_k)$

$$Q_{d_k} = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, s) B(s) Q(s) B^T(s) \Phi(t_{k+1}, s)^T ds \tag{38}$$

A good approximation is: $Q_d = BQB^T \Delta T$ this is only good as long as the eigenvalue norm satisfies $\|A\Delta t\|_F << 1$

Covariance matrices

- Let $x = [x(1), x(2), ..., x(n)] \in \mathbb{R}^n$, with $x(i)$ the i^{th} component of x, then the covariance matrix $P \in \mathbb{R}^{n \times n}$ is defined as:

$$\begin{aligned} \mathrm{COV}(x) &\triangleq \mathbf{E}[(x - \bar{x})(x - \bar{x})^T] \\ &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} (x - \bar{x})(x - \bar{x})^T dx(1)dx(2)\cdots dx(n) \quad \triangleq P \end{aligned}$$

- The i^{th} diagonal elements of P are the variance of $x(i)$ given by σ_i^2
- The off-diagonals are the cross-correlations, given by $\sigma_i \sigma_j$
- The covariance matrix is symmetric and positive definite, it can always be diagonalized

Noise properties

Additive process and measurement noise

The zero-mean additive Gaussian white noise assumption
$\omega_k \sim \mathbb{N}(0, Q_k)$: process noise, encodes the uncertainty in the sensors
$v_k \sim \mathbb{N}(0, R_k)$: The measurement noise, encodes our uncertainty in modeling the process

- Here $Q_k \in \mathbb{R}^{n \times n}$ and $R_k \in \mathbb{R}^{l \times l}$ are positive definite matrices, encoding the process and measurement noise covariances
- Typically sufficient to pick diagonal matrices with positive entries
- The measurement noise R_k (typically stationary: R) is typically provided in the sensor specification sheets, its the variance of the sensor
- Q_k (or when stationary: Q) is a little more difficult to find, typically this is the variable that needs to be *tuned*

Some notation

Kalman filter has two stages:

- Prediction: This is the a-priori stage, that is before the measurement is available, or in between any two measurements. We will denote all variables in this stage with a $-$ sign. E.g. P^-, $\hat{x}^-$
- Correction: This is the posteriori stage, that is after the measurement is available. We will denote all variables in this stage with a $+$ sign, E.g. P^+, $\hat{x}^+$
- We study first the discrete implementation of the Kalman filter, so k for us is the time or iteration step

Mean and Covariance propagation discrete time

$$E[x_{k+1}] = E[\Phi_k x_k + \omega_k] \tag{39}$$
$$E[x_{k+1}] = \Phi_k E[x_k] + 0 \tag{40}$$

Let $\mu_k = E[x_k]$ Covariance propagation

$$P_k = E[(x_k - \mu_k)(x_k - \mu_k)^T] \tag{41}$$

Hence

$$P_{k+1} = E[(x_{k+1} - \mu_{k+1})(x_{k+1} - \mu_{k+1})^T] \tag{42}$$
$$= E[(\Phi_k(x_k - \mu_k) + \omega_k)(\Phi_k(x_k - \mu_k) + \omega_k)^T] \tag{43}$$
$$= E[\Phi_k(x_k - \mu_k)(x_k - \mu_k)^T\Phi_k^T + \omega_k\omega_k^T + \Phi_k(x_k - \mu_k)\omega_k^T \tag{44}$$
$$+ \omega_k(x_k - \mu_k)^T\Phi_k^T]$$
$$P_{k+1} = \Phi_k P_k \Phi_k^T + Q_k \tag{45}$$

where $E[w_k w_k^T] = Q_k$ and noting that $E[\omega_k] = 0$

So the prediction is

$$\hat{x}_k^- = \phi_k \hat{x}_{k-1}^+ \quad (46)$$

$$P_{k+1}^- = \Phi_k P_k^+ \Phi_k^T + Q_k \quad (47)$$

Remember we are propagating the mean $\hat{x}_k$

So the prediction is

$$\hat{x}_k^- = \phi_k \hat{x}_{k-1}^+ \quad (46)$$

$$P_{k+1}^- = \Phi_k P_k^+ \Phi_k^T + Q_k \quad (47)$$

Remember we are propagating the mean $\hat{x}_k$ Now we use the observer equation to set up the correction

$$\hat{x}_{k+1}^+ = \Phi_k x_k + L_k(y_k - H_k \hat{x}_k^-) \quad (48)$$

So now

$$\hat{x}_k^+ = \hat{x}_k^- + L_k \delta y_k^- \quad (49)$$

Innovation term

$$\delta y_k^- = y_k - y_k^- = y_k - H_k \hat{x}_k^-$$

Now the prediction error is $e_k^- = x_k - \hat{x}_k^-$ (a-priori error) Let $e_k^+ = x_k - \hat{x}_k^+$ be the correction (a-posteriori) error and the **innovation term**:

$$\delta y_k^- = H_k \delta x_k^- + v_k \tag{50}$$

Now

$$\begin{aligned} e_k^+ &= x_k - \hat{x}_k^+ && (51) \\ &= x_k - \hat{x}_k^- - L_k(\delta y_k^-) && (52) \end{aligned}$$

Now the prediction error is $e_k^- = x_k - \hat{x}_k^-$ (a-priori error) Let $e_k^+ = x_k - \hat{x}_k^+$ be the correction (a-posteriori) error and the **innovation term**:

$$\delta y_k^- = H_k \delta x_k^- + v_k \tag{50}$$

Now

$$\begin{aligned} e_k^+ &= x_k - \hat{x}_k^+ && (51)\\ &= x_k - \hat{x}_k^- - L_k(\delta y_k^-) && (52)\\ &= x_k - \hat{x}_k^- - L_k(H_k e_k^- + v_k) && (53) \end{aligned}$$

Now the prediction error is $e_k^- = x_k - \hat{x}_k^-$ (a-priori error) Let $e_k^+ = x_k - \hat{x}_k^+$ be the correction (a-posteriori) error and the **innovation term**:

$$\delta y_k^- = H_k \delta x_k^- + v_k \tag{50}$$

Now

$$\begin{aligned} e_k^+ &= x_k - \hat{x}_k^+ && (51)\\ &= x_k - \hat{x}_k^- - L_k(\delta y_k^-) && (52)\\ &= x_k - \hat{x}_k^- - L_k(H_k e_k^- + v_k) && (53)\\ &= e_k^- - L_k(H_k e_k^- + v_k) && (54)\\ e_k^+ &= (I - L_k H_k) e_k^- - L_k v_k && (55) \end{aligned}$$

$$e_k^+ = (I - L_k H_k) e_k^- - L_k v_k \tag{56}$$

Let us now derive P_k^+

$$P_k^+ = E[e_k^+ {e_k^+}^T] \tag{57}$$

$$e_k^+ = (I - L_k H_k) e_k^- - L_k v_k \tag{56}$$

Let us now derive P_k^+

$$\begin{aligned} P_k^+ &= E[e_k^+ {e_k^+}^T] && (57)\\ &= E[(I - L_k H_k) e_k^- - L_k v_k)(I - L_k H_k) e_k^- - L_k v_k)^T] && (58) \end{aligned}$$

$$e_k^+ = (I - L_k H_k) e_k^- - L_k v_k \tag{56}$$

Let us now derive P_k^+

$$\begin{aligned} P_k^+ &= E[e_k^+ {e_k^+}^T] && (57)\\ &= E[(I - L_k H_k) e_k^- - L_k v_k)(I - L_k H_k) e_k^- - L_k v_k)^T] && (58)\\ &= E[(I - L_k H_k) e_k^- {e_k^-}^T (I - L_k H_k)^T + L_k R_k L_k^T] && (59) \end{aligned}$$

Recapping: Prediction:

$$\begin{aligned} \hat{x}_k^- &= \phi_k \hat{x}_{k-1}^+ \\ P_{k+1}^- &= \Phi_k P_k^+ \Phi_k^T + Q_k \end{aligned}$$

Recapping: Prediction:

$$\begin{aligned} \hat{x}_k^- &= \phi_k \hat{x}_{k-1}^+ \\ P_{k+1}^- &= \Phi_k P_k^+ \Phi_k^T + Q_k \end{aligned}$$

Correction after measurement:

$$\hat{x}_k^+ = \hat{x}_k^- + L_k \delta y_k^-$$

Posterior covariance

$$P_k^+ = (I - L_k H_k) P_k^- (I - L_k H_k)^T + L_k R_k L_k^T \tag{60}$$

Recapping: Prediction:

$$\begin{aligned} \hat{x}_k^- &= \phi_k \hat{x}_{k-1}^+ \\ P_{k+1}^- &= \Phi_k P_k^+ \Phi_k^T + Q_k \end{aligned}$$

Correction after measurement:

$$\hat{x}_k^+ = \hat{x}_k^- + L_k \delta y_k^-$$

Posterior covariance

$$P_k^+ = (I - L_k H_k) P_k^- (I - L_k H_k)^T + L_k R_k L_k^T \tag{60}$$

Now what we need is the optimal gain L_k

The optimal Kalman gain

- What does optimal mean here: What if we chose the gain to minimze the error variance!

The optimal Kalman gain

- What does optimal mean here: What if we chose the gain to minimze the error variance!
- The error variance is contained in the diagonal of P_k^+

Optimal gain L_k

To compute the optimal gain L_k we minimize $trace(P_k^+)$ wrt L_k. To do this, solve $\frac{\partial trace(P_k^+)}{\partial L_k} = 0$ for L_k

Remember trace is the sum of the diagonal elements in the matrix

Computing the optimal gain

Multiply the terms out and drop the subscript k for convinience

$$P^+ = P^- - LHP^- - P^- H^T L^T + L(HP^- H^T + R)L^T \tag{61}$$

Computing the optimal gain

Multiply the terms out and drop the subscript k for convinience

$$P^+ = P^- - LHP^- - P^- H^T L^T + L(HP^- H^T + R)L^T \tag{61}$$

Now taking the trace, and remembering $trace(AB) = trace(BA)$ and $trace(A^T) = trace(A)$, and P^- is symmetric positive definite Hence

$$trace(P^+) = trace(P^-) - 2trace(LHP^-) + trace[L(HP^- H^T + R)L^T] \tag{62}$$

Now taking the derivative $\frac{\partial trace(P_k^+)}{\partial L_k}$

$$\frac{\partial trace(P_k^+)}{\partial L_k} = -2(HP^-)^T + 2L(HP^- H^T + R) \tag{63}$$

For minimum, set $\frac{\partial trace(P_k^+)}{\partial L_k} = 0$,

For minimum, set $\frac{\partial trace(P_k^+)}{\partial L_k} = 0$,

Now remember $\frac{\partial(trace(AB))}{\partial A} = B^T$ if AB is square, and $\frac{\partial(trace(ABA^T))}{\partial A} = 2AB^T$ if B is symmetric. So now

For minimum, set $\frac{\partial trace(P_k^+)}{\partial L_k} = 0$,

Now remember $\frac{\partial(trace(AB))}{\partial A} = B^T$ if AB is square, and $\frac{\partial(trace(ABA^T))}{\partial A} = 2AB^T$ if B is symmetric. So now

$$-2(HP^-)^T + 2L(HP^-H^T + R) = 0 \tag{64}$$

For minimum, set $\frac{\partial trace(P_k^+)}{\partial L_k} = 0$,

Now remember $\frac{\partial(trace(AB))}{\partial A} = B^T$ if AB is square, and $\frac{\partial(trace(ABA^T))}{\partial A} = 2AB^T$ if B is symmetric. So now

$$-2(HP^-)^T + 2L(HP^-H^T + R) = 0 \tag{64}$$

Hence

Kalman Gain

$$L = P^-H^T(HP^-H^T + R)^{-1} \tag{65}$$

The Kalman Filtering Algorithm

- The Kalman filter has two steps:
- Prediction step:
  - In this step we predict forward the state of the system using our model and Q
  - This is our best guess of what the system state would look like
  - But it will deviate from the true state if the system evolves in a different manner than we expecte
- Correction step: To ensure that our predictions do not drift for too long, the KF utilizes the idea of feedback corrections
  - In this step we correct our predicted state using feedback between predicted measurement and the actual sensor measurement
  - The correction brings our prediction back on track, without having to have information about all the states, or doing it all the time
- Together the predict-correct framework leads to a robust state estimation technique

Kalman filtering algorithm mathematical specifics

Initialize $x_0 \sim \mathbb{N}(0, P_0)$

Prediction step

$$x_k^- = \Phi_k x_{k-1}$$
$$P_k^- = \Phi_k P_{k-1} \Phi_k^T + Q_k$$

Correction step

$$e_k = y_k - H_k x_k^-$$
$$S_k = H_k P_k^- H_k^T + R_k$$
$$L_k = P_k^- H_k^T S_k^{-1}$$
$$x_k^+ = x_k^- + L_k e_k$$
$$P_k^+ = P_k^- - L_k S_k L_k^T$$

Kalman filter algorithm

If we assume that Φ_k, H_k, Q_k, R_k do not change, we can re-write the algorithm more simply:

KF algorithm

Prediction step

$$x_k^- = \Phi x_{k-1}$$
$$P_k^- = \Phi P_{k-1} \Phi^T + Q$$

Correction step

$$L_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad (66)$$
$$S_k = H P_k^- H^T + R \quad (67)$$
$$x_k = x_k^- + L_k (y_k - H x_k^-) \quad (68)$$
$$P_k = P_k^- - L_k S_k L_k^T \quad (69)$$

- Note that P_k does not depend on x_k, this is a direct consequence of the linearity and Gaussian noise assumption
- This means we can pre-compute K_k off line by iteratively solving (1) and (2) until they converge

Continuous time Kalman Filter

If we assume that A, C, Q, R are continuous-time counterparts:

KF algorithm

Prediction step Initialize $\hat{x}^-(t) = \hat{x}(t = k), P(t) = P_k(t = k)$

$$\dot{\hat{x}}^- = A\hat{x}$$
$$\dot{P}^- = AP + PA^T + Q$$

Correction step

$$P_k^- = P(t = k) \tag{70}$$
$$L_k = P_k^- C^T (CP_k^- C^T + R)^{-1} \tag{71}$$
$$x_k = x_k^- + L(y_k - Cx_k^-) \tag{72}$$
$$P_k = [I - L_k C]P^-(k) \tag{73}$$

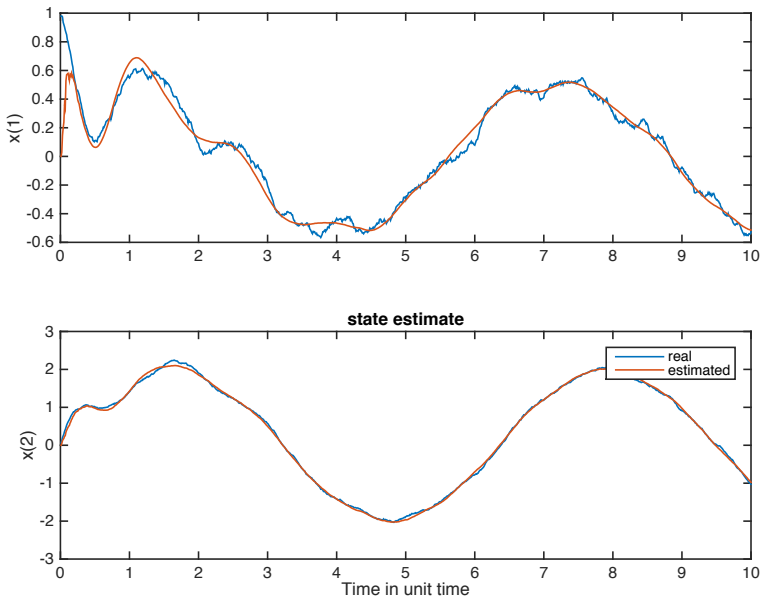
- Note that P_k does not depend on x_k, this is a direct consequence of the linearity and Gaussian noise assumption
- This means we can pre-compute K_k off line by iteratively solving (1) and (2) until they converge

Outline

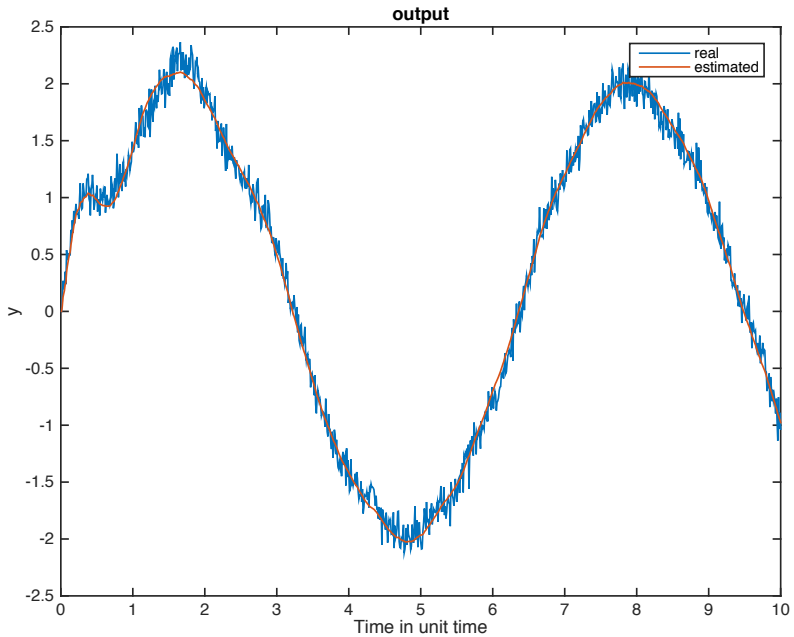
Software example

$$A = \begin{bmatrix} -1 & -5 \\ 6 & -1 \end{bmatrix}$$
$$C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

- Matlab code `KF_simple.m`
- The measurement noise variance is 0.1 for both states
- The process noise variance is 0.01 for both states



(a) States



(b) Output

More details on the software implementation

- The software implementation use a slightly different notation for ease in variable naming
- x_k^- is termed $\tilde{x}$, P_k^- is termed $\tilde{P}$
- The true state of the system (required for comparison) is simulated, and called x_k, whereas the estimated state is called $\hat{x}_k$ consistent with estimation theory notation
- The system also has a known input $u(t) \in \mathbb{R}$ and is assumed to be continuous, that is:

$$\dot{x} = Ax + Bu + \zeta$$

- With $B = [10]^T$
- The input is sinusoidal, and the system is discretized to work with the framework outlined in class

What if the noise is non-Gaussian or the system is non-linear?

- If the system dynamics are non-linear but sufficiently smooth, then we can try local linearization
- This leads to the Extended Kalman Filter (more about this next week)
- If the dynamics are not sufficiently smooth, or the noise is non-Gaussian, we can utilize Particle Filters (more about this next week)
- The idea here is to create a cloud of particles, transform them through the dynamics and noise, and then re-compute the mean and variance at the other side
- A smart way of doing this is known as the Unscented Kalman Filter (Julier and Uhlmann, 1997)

Extended Kalman Filter

What if the dynamics are nonlinear?

$$\dot{x} = f(x, u) \tag{74}$$

$$y = h(x) \tag{75}$$

The Bayesian inference problem will in general be intractable (remember Champan Komlogrov equation and the challenges with Bayesian inference)
Our options:

- Full-blown sample-based Bayesian inference (Markov Chain Monte Carlo (see ABE 598))
- Particle filters: Same as above, but with a specifically selected set of sample "particles"
- Extended Kalman Filter: Linearized filter in a nonlinear setting (as opposed to a fully linearized filter)

EKF algorithm

Initialize $\hat{x}_0 = x(0)$, $P_0 = diag([\sigma_{x_1}^2, \sigma_{x_2}^2, \ldots, \sigma_{x_n}^2])$

Prediction step

$$\hat{x}_k^- = \int_t^{t+\Delta t} f(\hat{x}_{k-1}, u_{k-1})dt$$

$$A_k = \frac{\partial f(x,u)}{\partial x}|_{x=\hat{x}_{k-1}}; \quad \Phi_k = e^{(A_k \Delta t)}$$

$$P_k^- = \Phi_k P_{k-1} \Phi_k^T + Q$$

Correction step

$$y(k) = h(x) \tag{76}$$

$$H_k = \frac{\partial h(x)}{\partial x}|_{x=\hat{x}k-1} \tag{77}$$

$$L_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{78}$$

$$S_k = H P_k^- H^T + R \tag{79}$$

$$x_k = x_k^- + L(y_k - H x_k^-) \tag{80}$$

$$P_k = P_k^- - L S_k L^T \tag{81}$$

Outline

Point mass system

Consider a point mass system

$$\dot{p}(t) = v(t), \tag{82}$$

$$\dot{v}(t) = a(t) \tag{83}$$

The output of the accelerometer is the true acceleration, plus bias, plus noise. So the measurement model for the accelerometer is:

$$\tilde{u}(t) = \underbrace{a(t)}_{\text{true acceleration}} - \underbrace{b(t)}_{bias} - \underbrace{v_a(t)}_{noise(white)} \tag{84}$$

Point mass system

Consider a point mass system

$$\dot{p}(t) = v(t), \tag{82}$$
$$\dot{v}(t) = a(t) \tag{83}$$

The output of the accelerometer is the true acceleration, plus bias, plus noise. So the measurement model for the accelerometer is:

$$\tilde{u}(t) = \underbrace{a(t)}_{\text{true acceleration}} - \underbrace{b(t)}_{bias} - \underbrace{v_a(t)}_{noise(white)} \tag{84}$$

We will assume that bias is a slowly evolving process that is stationary, so

$$\dot{b}(t) = \omega_b(t)$$

•Where, $\omega_b \sim N(0, \sigma_{\omega_b})$. Let's say that $\sigma_b^2 = 1.0 \times 10^{-6} m^2/s^2$. We will also assume that the initial bias is a draw from a Gaussian distribution: $b(0) \sim N(0, \sigma_{\omega_b})$

Point mass system

Consider a point mass system

$$\dot{p}(t) = v(t), \tag{82}$$

$$\dot{v}(t) = a(t) \tag{83}$$

The output of the accelerometer is the true acceleration, plus bias, plus noise. So the measurement model for the accelerometer is:

$$\tilde{u}(t) = \underbrace{a(t)}_{\text{true acceleration}} - \underbrace{b(t)}_{bias} - \underbrace{v_a(t)}_{noise(white)} \tag{84}$$

We will assume that bias is a slowly evolving process that is stationary, so

$$\dot{b}(t) = \omega_b(t)$$

•Where, $\omega_b \sim N(0, \sigma_{\omega_b})$. Let's say that $\sigma_b^2 = 1.0 \times 10^{-6} m^2/s^2$. We will also assume that the initial bias is a draw from a Gaussian distribution: $b(0) \sim N(0, \sigma_{\omega_b})$

•v_a is Gaussian white note, $v_a(t) \sim N(0, \sigma_{v_a})$, let's say $\sigma_{v_a}^2 = 2.5 \times 10^{-3} m^2/s^3$

How do the state estimates behave

Observer (navigation or prediction or mechanization) equations from 82

$$\dot{\hat{p}}(t) = \hat{v}(t), \tag{85}$$

$$\dot{\hat{v}}(t) = \hat{a}(t) \tag{86}$$

Where, from (84)

$$\hat{a}(t) = \underbrace{\tilde{u}(t)}_{accelerometermeasurements} + \underbrace{\hat{b}(t)}_{KFestimateofbias}$$

$$\hat{a}(t) = \underbrace{a(t)}_{\text{true acceleration}} - \underbrace{b(t)}_{bias} - \underbrace{v_a(t)}_{noise(white)} + \underbrace{\hat{b}(t)}_{KFestimateofbias} \approx a(t) - v_a(t)$$

What should $\dot{\hat{b}}(t)$ be?

How do the state estimates behave

Observer (navigation or prediction or mechanization) equations from 82

$$\dot{\hat{p}}(t) = \hat{v}(t), \tag{85}$$
$$\dot{\hat{v}}(t) = \hat{a}(t) \tag{86}$$

Where, from (84)

$$\hat{a}(t) = \underbrace{\tilde{u}(t)}_{accelerometermeasurements} + \underbrace{\hat{b}(t)}_{KFestimateofbias}$$

$$\hat{a}(t) = \underbrace{a(t)}_{\text{true acceleration}} - \underbrace{b(t)}_{bias} - \underbrace{v_a(t)}_{noise(white)} + \underbrace{\hat{b}(t)}_{KFestimateofbias} \approx a(t) - v_a(t)$$

What should $\dot{\hat{b}}(t)$ be? Remember that $E[\dot{b}(t)] = 0$ because ω_b is Gaussian white noise, so, we have our third equation

$$\dot{\hat{b}}(t) = 0 \tag{87}$$

Putting it all together

Our filter mechanization equations:

$$\begin{bmatrix} \dot{\hat{p}} \\ \dot{\hat{v}} \\ \dot{\hat{b}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{p} \\ \hat{v} \\ \hat{b} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{u} \\ \omega_b \end{bmatrix} \tag{88}$$

Note that we are using the accelerometer measurements $\tilde{u}$ as a surrogate "control" input

Putting it all together

Our filter mechanization equations:

$$\begin{bmatrix} \dot{\hat{p}} \\ \dot{\hat{v}} \\ \dot{\hat{b}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{p} \\ \hat{v} \\ \hat{b} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{u} \\ \omega_b \end{bmatrix} \tag{88}$$

Note that we are using the accelerometer measurements $\tilde{u}$ as a surrogate "control" input
Now we need our measurement equations, let's assume position is being measured, then

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{p} \\ \hat{v} \\ \hat{b} \end{bmatrix} \tag{89}$$

Now we have enough to do our Kalman Filtering, but will the filter converge? Will we get an optimal gain matrix L?

Observability analysis

We need to check observability from (24). Use matlab command obsv(A,C).
The rank needs to be 3 for observability
First case, only position measured

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{90}$$

Observability analysis

We need to check observability from (24). Use matlab command obsv(A,C).
The rank needs to be 3 for observability
First case, only position measured

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{90}$$

Rank is 3, system is observable
Second case, only velocity measured.

$$C = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{91}$$

Observability analysis

We need to check observability from (24). Use matlab command obsv(A,C).
The rank needs to be 3 for observability
First case, only position measured

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{90}$$

Rank is 3, system is observable
Second case, only velocity measured.

$$C = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{91}$$

Rank is 2, system is NOT observable
Third case, both position and velocity measured

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{92}$$

Observability analysis

We need to check observability from (24). Use matlab command obsv(A,C).
The rank needs to be 3 for observability
First case, only position measured

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{90}$$

Rank is 3, system is observable
Second case, only velocity measured.

$$C = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{91}$$

Rank is 2, system is NOT observable
Third case, both position and velocity measured

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{92}$$

Rank is 3, system is observable

Outline

GPS+Encoders

Now we will consider designing a navigation system for a ground robot that has GPS and encoders. We will use encoders to mechanize the (extended) Kalman filter and use GPS to correct it.
Kimenatic model of the Dubin's car robot from Module 2, assuming no slip and no lateral velocity in the body frame

$$\dot{x} = u \ \sin\psi \tag{93}$$
$$\dot{y} = u \ \cos\psi \tag{94}$$
$$\dot{\psi} = \omega \tag{95}$$

Let $v^b = \begin{bmatrix} u \\ v \end{bmatrix}$. The velocity vector in the tangent frame is

$$v^t = R_{b\to t} v^b$$

Where $R_{b\to t} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}$

The encoders

The encoders U_L and U_R measure left and right encoder velocities

$$u = \frac{1}{2}(U_L + U_R) \tag{96}$$

$$\omega = \frac{1}{L}(U_L - U_R) \tag{97}$$

The encoders

The encoders U_L and U_R measure left and right encoder velocities

$$u = \frac{1}{2}(U_L + U_R) \tag{96}$$

$$\omega = \frac{1}{L}(U_L - U_R) \tag{97}$$

We use the above to mechanize equations (93), to correct, we use GPS, which we assume is at the center of the robot (more on this later), and measures position

$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \omega \end{bmatrix} \tag{98}$$

Equations (93) are nonlinear, so we need here an Extended Kalman Filter, remember $A_k = \frac{\partial f}{\partial x}|_{x=\hat{x}}$
Doing the math, we get

$$\mathscr{A}_k = \begin{bmatrix} 0 & 0 & -u\sin\psi \\ 0 & 0 & u\cos\psi \\ 0 & 0 & 0 \end{bmatrix} \tag{99}$$

and $\Phi_k = e^{A\Delta T}$
So, the linearized prediction model is

$$\hat{x}_{k+1} = \Phi_k \hat{x}_k + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega$$

Outline

Attitude Hold and Reference System

- Used in aircraft that do not have GPS to hold attitude, does not work in the presence of strong acceleration
- Use inertial sensor only + magnetometers
- 3 axis accelerometers, gyroscopes, and magnetometers
- Idea: Integrate gyro outputs to predict attitude and then correct with gravity direction sensed from accelerometers

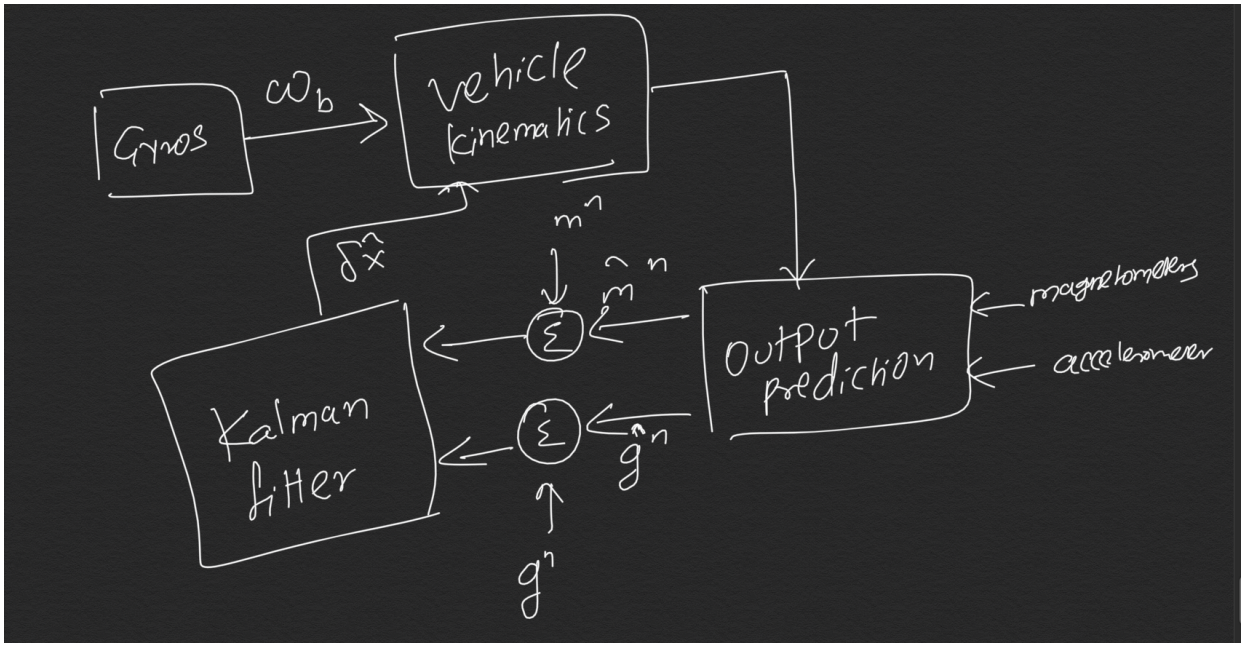


Figure: AHRS schematic

AHRS mechanization

- $\omega_{b/I}$ Angular rate of body WRT inertial
- $\omega_{b/N}$ Angular rate of body WRT body carried navigation frame (remember Module 2)

$$\omega_{N/b} = \omega_{N/I} - \omega_{b/I}$$

The above is a 3-dimensional vector. Let $\bar{q}$ denote the rotational quaternion from nav frame to body frame $\bar{q} = [q_0, q_1, q_2, q_3]$, then

$$\dot{q} = \frac{1}{2}\tilde{q}\omega_{n/b}$$

Where $\tilde{q}$ is the quaternion multiplication operator, remember Module 2. Now the body force vector is

$$f_{b/I} = a_{b/I} - g_{b/I} \tag{100}$$

and

$$g_{b/I} = R_{I\to b}\begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{bmatrix} g$$

Sensor model $u = \omega_{b/I} + \underbrace{x_g}_{\text{bias}} + \underbrace{v_g}_{\text{noise}}$

Outline

Example: GPS-INS integration

States to estimate:

- p: x, y, z position in NED frame (which we are going to assume is inertial)
- v: U, V, W velocities in NED frame
- q: Attitude quaternion (note the discussion on error quaternion in a latter slide)
- b_ω: Bias of the rate gyro
- b_a: Bias of the accelerometer

Using the following measurements:

- $\tilde{a}$: 3 axis accelerometers
- $\tilde{\omega}$: 3 axis gyro
- $\tilde{m}$: 3 axis magnetometer (we won't really use this)
- $\tilde{p}$: position from GPS
- $\tilde{v}$: velocity from GPS

INS mechanization equations

Now we note the kinematic equations of a rigid body moving in 3-dimensional space
They are:

$$ {}^{I}\dot{\hat{p}} = {}^{I}\hat{v} \tag{101} $$

$$ {}^{I}\dot{\hat{v}} = \hat{R}_{b\to I}({}^{b}a - \hat{b}_a) \tag{102} $$

$$ \dot{\hat{q}} = -\frac{1}{2}\Omega(\omega)q \tag{103} $$

$$ \dot{\hat{b}}_\omega = 0 \tag{104} $$

$$ \dot{\hat{b}}_a = 0 \tag{105} $$

$$ \tag{106} $$

$$ \Omega(\omega) = \begin{bmatrix} 0 & P & Q & R \\ -P & 0 & -R & Q \\ -Q & R & 0 & -P \\ -R & -Q & P & 0 \end{bmatrix} \tag{107} $$

Inertial measurement models

•Acceleration

$$\tilde{a} = a - b_a \tag{108}$$

•Angular rates, $\omega = [P, Q, R]^T$, and the measurement model for the gyroscopes:

$$\tilde{\omega} = \omega - b_\omega \tag{109}$$

where $\tilde{\omega} = [p, q, r]$,
So, $P = p + b_{\omega_p}; Q = q + b_{\omega_q}; R = r + b_{\omega_r}$

Attitude quaternion

Let the quaternion denoting the rotation from body to inertial frame $q = [q_1, q_2, q_3, q_4]$, where q_1 is the scalar part denoting the rotation and $[q_2, q_3, q_4]$ is the axis of rotation

$$R_{b\to I} = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2q_3 + q_1q_4) & 2(q_2q_4 - q_1q_3) \\ 2(q_2q_3 - q_1q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2(q_3q_4 + q_1q_2) \\ 2(q_2q_4 + q_1q_3) & 2(q_3q_4 - q_1q_2) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix} \tag{110}$$

And,

$$R_{I\to b} = R_{b\to I}^T \tag{111}$$

•Don't forget to normalize the quaternion in your code every time-step, you do this by setting $q \leftarrow \frac{q}{norm(q)}$

Linearization

$$F_{pv} = \frac{\partial^I \dot{\hat{p}}}{\partial^I \hat{v}} = I_{3\times 3} \tag{112}$$

$$F_{vq} = \frac{\partial^I \dot{\hat{v}}}{\partial^I \hat{q}} = \frac{\partial R_{b\to I}\,{}^b a}{\partial q} \tag{113}$$

$$F_{vb_a} = \frac{\partial^I \dot{\hat{v}}}{\partial^I \hat{b}_a} = -R_{b\to I} \tag{114}$$

$$F_{qq} = \frac{\partial^I \dot{\hat{q}}}{\partial^I \hat{q}} = -\frac{1}{2}\Omega(\omega) \tag{115}$$

$$F_{qb_\omega} = \frac{\partial^I \dot{\hat{q}}}{\partial \hat{b}_\omega} \tag{116}$$

Linearization

Let us look at some of these equations in more detail

$$\frac{\partial^I \dot{\hat{v}}}{\partial^I \hat{q}} = \frac{\partial R_{b\to I}{}^I a}{\partial q} \tag{117}$$

$$\frac{\partial^I \dot{\hat{v}}}{\partial^I \hat{q}} = \begin{bmatrix} 2(q_1 a_x - q_4 a_y + q_3 a_z) & 2(q_2 a_x + q_3 a_y + q_4 a_z) & 2(-q_3 a_x + q_2 a_y + q_1 a_z) & 2(-q_4 a_x - q_1 a_y + q_2 a_z) \\ 2(q_4 a_x + q_1 a_y - q_2 a_z) & 2(q_3 a_x - q_2 a_y - q_1 a_z) & 2(q_2 a_x + q_3 a_y + q_4 a_z) & 2(q_1 a_x - q_4 a_y + q_3 a_z) \\ 2(-q_3 a_x + q_2 a_y + q_1 a_z) & 2(q_4 a_x + q_1 a_y - q_2 a_z) & 2(-q_1 a_x + q_4 a_y - q_3 a_z) & 2(q_2 a_x + q_3 a_y + q_4 a_z) \end{bmatrix} \tag{118}$$

$$\frac{\partial^I \dot{\hat{q}}}{\partial \hat{b}_\omega} = -0.5 \frac{\partial \Omega(\omega) q}{\partial b_\omega} \tag{119}$$

$$= \frac{1}{2} \begin{bmatrix} q_2 & q_3 & q_4 \\ -q_1 & q_4 & -q_3 \\ -q_4 & -q_1 & q_2 \\ q_3 & -q_2 & -q_1 \end{bmatrix} \tag{120}$$

Putting together the linearized transition matrix

Here Z is a zero matrix, and I is the identity matrix

$$A = \begin{bmatrix} Z(3\times 3) & I(3\times 3) & Z(3\times 4) & Z(3\times 3) & Z(3\times 3) \\ Z(3\times 3) & Z(3\times 3) & F_{vq} & Z(3\times 3) & F_{vb_a} \\ Z(4\times 3) & Z(4\times 3) & F_{qq} & F_{qb_\omega} & Z(4\times 3) \\ Z(3\times 3) & Z(3\times 3) & Z(3\times 4) & Z(3\times 3) & Z(3\times 3) \\ Z(3\times 3) & Z(3\times 3) & Z(3\times 4) & Z(3\times 3) & Z(3\times 3) \end{bmatrix} \quad (121)$$

•Don't forget to discretize A. If you want to use continuous A, i.e. 121 then use $\dot{P} = AP + PA^T + Q$

Measurements

Our measurements are GPS position and velocity, so $z = [x, y, z, v_x, v_y, v_z]^T$ Where z contains positions and velocities at the CG. However, the GPS is mounted offset from the CG (this is very important) at the location r_{GPS}, where r is in the body fixed frame, so

$$p_{CG} = p_{GPS} - R_{b \to I} r_{GPS} \tag{122}$$

$$v_{CG} = v_{GPS} - R_{b \to I} \omega \times r_{GPS} \tag{123}$$

Remember $R_{b \to I}$ is a function of the quaternions, so we need to linearize this to get our H matrix. Now $r_{GPS} = [1.5, 0, 0]^T$ for the dataset we are using. So we can ignore the second and third column of $R_{b \to I}$

Linearizing the measurement model

$$H_{xq} = \begin{bmatrix} -r_{GPS}(1)2q_1 & -r_{GPS}(1)2q_2 & r_{GPS}(1)2q_3 & r_{GPS}(1)2q_4 \\ -r_{GPS}(1)2q_4 & -r_{GPS}(1)2q_3 & -r_{GPS}(1)2q_2 & -r_{GPS}(1)2q_1 \\ r_{GPS}(1)2q_3 & -r_{GPS}(1)2q_4 & r_{GPS}(1)2q_1 & -r_{GPS}(1)2q_2 \end{bmatrix} \tag{124}$$

$$H_{vq} = \begin{bmatrix} r_{GPS}(1)2q_3Q + r_{GPS}(1)2q_4R & r_{GPS}(1)2q_4Q - r_{GPS}(1)2q_3R & r_{GPS}(1)2q_1Q - r_{GPS}(1)2q_2R & r_{GPS}(1)2q_2Q + r_{GPS}(1)2q_1R \\ -r_{GPS}(1)2q_2Q - r_{GPS}(1)2q_1R & r_{GPS}(1)2q_2R - r_{GPS}(1)2q_1Q & r_{GPS}(1)2q_4Q - r_{GPS}(1)2q_3R & r_{GPS}(1)2q_3Q + r_{GPS}(1)2q_4R \\ r_{GPS}(1)2q_1Q - r_{GPS}(1)2q_2R & -r_{GPS}(1)2q_2Q - r_{GPS}(1)2q_1R & -r_{GPS}(1)2q_3Q - r_{GPS}(1)2q_4R & r_{GPS}(1)2q_4Q - r_{GPS}(1)2q_3R \end{bmatrix} \tag{125}$$

$$H = \begin{bmatrix} I(3\times 3) & Z(3\times 3) & H_{xq} & Z(3\times 6) \\ Z(3\times 3) & I(3\times 3) & H_{vq} & Z(3\times 6) \end{bmatrix} \tag{126}$$

Error Quaternion

Quaternion: $q = [q_0, q_1, q_2, q_3]$ Define an error quaternion: $\delta q = [1, s]^T$, such that

$$\delta q \circ \hat{q} = q \tag{127}$$

In practice, this error quaternion takes on very small values, so its ok to say $\hat{s} = 0$ If we do this, we can simplify some calculations. See JFR paper by Chowdhary et al.

Outline

System Identification or Machine Learning

•The challenge here is to estimate a set of unknown parameters θ of the model $\dot{x} = f(x, \theta, u)$ using measurements $y = g(x, \theta, u)$
•This is still somewhat of an open problem, especially when f, g are nonlinear, and even when they are not, but g does not lead to full state observations.
We can formulate the problem as:

$$\dot{x} = f(x, \theta, u) \tag{128}$$
$$\dot{\theta} = 0 \tag{129}$$
$$y = f(x, \theta) \tag{130}$$

•This is a nonlinear equation, because of interaction with θ
•Need to solve as an Extended Kalman filter, or an UKF

HW 3 Q2

- To solve Q2, you need to simulate 2 different systems
- The first without noise, with the real a
- The second, with noise, with $\tilde{u}$

$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{b} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} p \\ v \\ b \end{bmatrix} \tag{131}$$

HW3 Q3

Now you have a measurement, and it is of the form

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \\ b \end{bmatrix} \tag{132}$$

When you do the correction step, make sure you do it every second.
The prediction should happen as fast as the inertial sensors give you data, assume 100Hz.

HW3 Q4

The states are now $[x, y, \theta, v, b]$ The linearized matrix is (before discretization)

$$A = \begin{bmatrix} 0 & 0 & -v\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & v\cos(\theta) & \sin(\theta) & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{133}$$

For the prediction step, integrate the continuous nonlinear dynamics first
For the correction step use the measurement matrix:

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ v \\ b \end{bmatrix} \tag{134}$$

The third measurement is the encoder

Observability

Observability tells us whether the measurements available (i.e. y) are sufficient to reconstruct the state (x).
For linear time invariant systems, this boils down to a simple condition on the matrix pair (C, A):

Observability condition

The pair (C, A) is observable if and only if the matrix $\mathscr{O}$ has full column rank

$$\mathscr{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{135}$$

Example of observability

Consider the HW system from Q3, the linearized matrices are given in 133 and the measurement matrix: 134
Let us use MATLAB to check the observability condition, try it for the following cases:

- $\theta = 45^0$, x, y measured
- $\theta = 45^0$, x, y, v measured
- $\theta = 0^0$, x, y measured
- $\theta = 90^0$, x, y measured

Example of observability

Consider the HW system from Q3, the linearized matrices are given in 133 and the measurement matrix: 134
Let us use MATLAB to check the observability condition, try it for the following cases:

- $\theta = 45^0$, x, y measured
- $\theta = 45^0$, x, y, v measured
- $\theta = 0^0$, x, y measured
- $\theta = 90^0$, x, y measured

Case study: estimation of velocity

Consider a situation where you have position measurements x at sample rate dt and you want to estimate velocity, $\dot{x}$ Formulate the problem as:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} = \zeta$$

Where ω is zero mean Gaussian white process noise and ζ is zero mean Gaussian white measurement noise. This is a second order random walk model, we are assuming that the acceleration is random, which isn't really true, but it works as an estimation model. You'll notice that the measurement y is nothing but noisy x.
To solve the estimation problem, simply design a Kalman filter with $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ and $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$ Its a bit tricky to tune it right, and you'll always have some delay in estimation of $\dot{x}$

Kalman Filters for navigation

Generic example: `https://www.youtube.com/watch?v=ymuhJ6pt52o`
Things to think about:

- What are navigation goals: mission goals, e.g. get from A to B, weed all crops, find life on another planet etc
- What kind of control and planning loop speeds are required? E.g. interceptive missile probably needs to have high control update rate (100Hz or more) than a slow submarine that's the meandering around looking for stuff
- What states of the robot are to be measured
- What sensors are available, exteroceptive, proprioceptive
- What are their update rates, noise characteristics and failure modes
- What sensors to use for prediction, and which for correction?
- Will we need a regular or a nonlinear Kalman filter?
- How to make sure the filter is robust?

Breakout sessions on navigation system design

- We will break-out into groups and discuss the design of navigation systems
- Guideline: think of proprioceptive sensors, exteroceptive sensors, which sensors to use for prediction, which for correction, what are their update rates (guess), how to make sure the Filter works.
- Group 1: Autonomous surface vessel
- Group 2: Autonomous submarine
- Group 3: Mars rover
- Group 4: USS Enterprise on intergalactic mission
- Group 5: TerraSentia under-canopy rover
- Group 6: A dirigible probe for sampling organic gases on Jupiter

Further Reading

- Gelb Arthur, **Applied Optimal Estimation**, MIT Press, Chapter 4: Optimal Linear Filtering
- Slides on the **Bayesian derivation of the Kalman Filtering equations** by Simo Särkkä: `http://becs.aalto.fi/~ssarkka/course_k2012/handout3.pdf`
- Simo's book: **Bayesian Filtering and Smoothing**, Chapte 4, available online: `http://becs.aalto.fi/~ssarkka/pub/cup_book_online_20131111.pdf`
- Christophersen, Henrik B., et al. "**A compact guidance, navigation, and control system for unmanned aerial vehicles.**" Journal of aerospace computing, information, and communication 3.5 (2006): 187-213.
- Chowdhary, Girish, et al. "**GPS denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft.**" Journal of Field Robotics 30.3 (2013): 415-438.