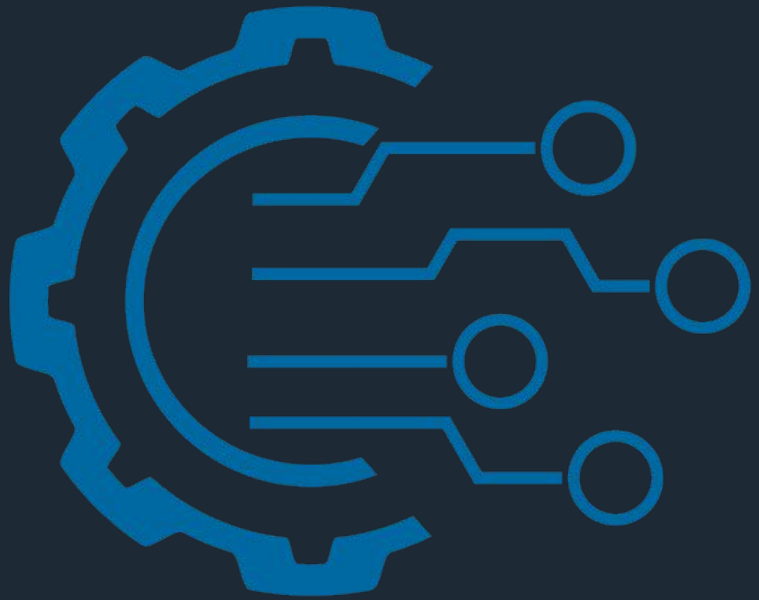


Technitute_

Data Fusion with the Linear Kalman Filter

Introduction Series

Theory and Implementation

Steven Dumble, PhD

Why Focus on Data Fusion and Kalman Filtering?

- Data Fusion is an amazing tool that is used pretty much in every modern piece of technology that involves any kind of sensing, measurement or automation.
- The Kalman Filter is one of the most widely used methods for data fusion. By understanding this process you will more easily understand more complicated methods.
- Difficult for beginners to comprehend how the filter works and how to apply the concepts in practice.
- So you don’t waste time trying to solve or debug problems that would be easily avoided with this knowledge! Become a **Subject Matter Expert**!

Who is this course for?

- University students or independent learners.
- Working Engineers and Scientists.
- Engineering professionals who wants to brush up on the math theory and skills related to data Fusion and Kalman filtering.
- Software Developers who wish to understand the basic concepts behind data fusion to aid in implementation or support of developing data fusion code.
- Anyone already proficient with the math “in theory” and want to learn how to implement the theory in code.

What is Covered?

- Basic Probability and Random Variables
- Dynamic Systems and State Space Representations
- Least Squares Estimation
- Linear Kalman Filtering
- Covers theory, implementation, use cases
- Theory explanation and analysis using Python and Simulations

Course Outcomes

By the end of this course you will know:

- How to probabilistically express uncertainty using probability distributions
- How to convert differential systems into a state space representation
- How to simulate and describe state space dynamic systems
- How to use Least Squares Estimation to solve estimation problems
- How to use the Linear Kalman Filter to solve optimal estimation problems
- How to derive the system matrices for the Kalman Filter in general for any problem
- How to optimally tune the Linear Kalman Filter for best performance
- How to implement the Linear Kalman Filter in Python

Steven Dumble, PhD

Technitute_

Welcome:
Course Outline

Steven Dumble, PhD

Technitute_

Course Outline

Data Fusion with the Linear Kalman Filter

- Welcome
- Introduction
- Probability
- Dynamic Systems
- Least Squares Estimation
- Linear Kalman Filter
  - Theory
  - Implementation
- Additional Examples
- Conclusion

Steven Dumble, PhD

Steven Dumble, PhD

Technitute_

Welcome:
Setting Up Python

Steven Dumble, PhD

Steven Dumble, PhD

Technitute_

Data Fusion with the Linear Kalman Filter

Introduction Series

Introduction

Steven Dumble, PhD

Introduction:
What is Data Fusion?

Steven Dumble, PhD

VIDEO HERE

Data Fusion is the process of integrating ***multiple*** data sources to form useful information that is more ***consistent*** and more ***accurate*** than the *original data sources*.

VIDEO HERE

This course covers a specific form of data fusion, call **sensor fusion** or **multi-sensor fusion**.

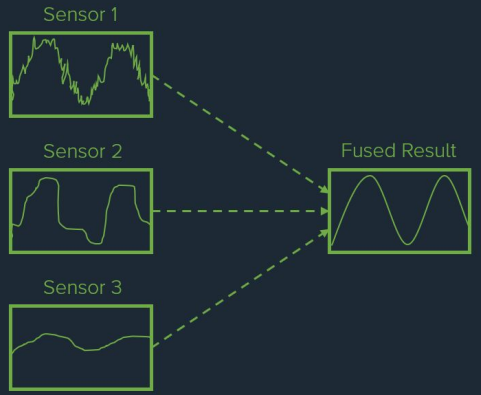
Combining sensor measurements together in such a way that the resulted fusion produces a **more confident** estimate than the individual measurements alone could provide.

Data Fusion

SHOW ANIMATION

Data Fusion

Fuse Multiple Sensors



Compensate For Sensor Errors

Technitute_

No sensor is perfect. Measurements are corrupted with errors such as **noise** and **biases**.

Steven Dumble, PhD

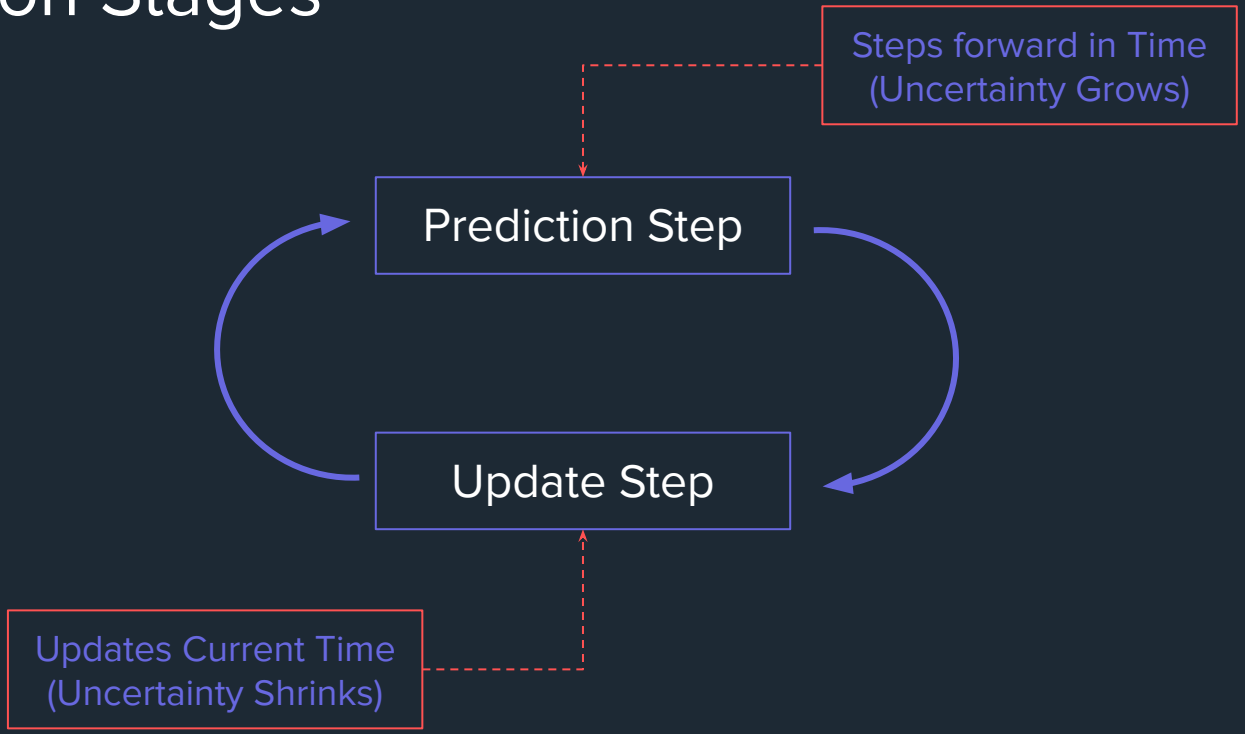
VIDEO HERE

Data Fusion

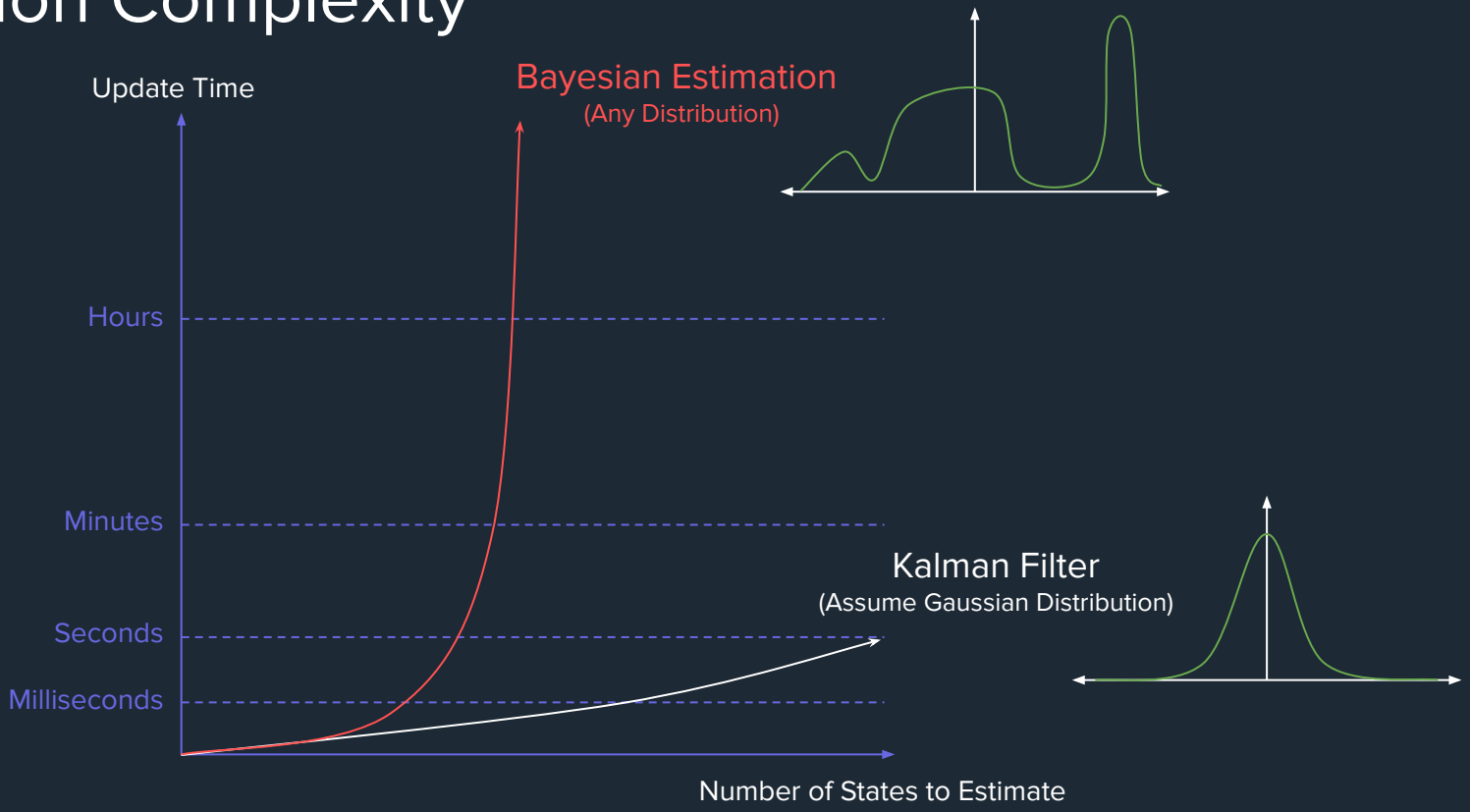
Introduction:
How does Sensor Fusion Work?

Steven Dumble, PhD

Data Fusion Stages



Computation Complexity



Key Ideas

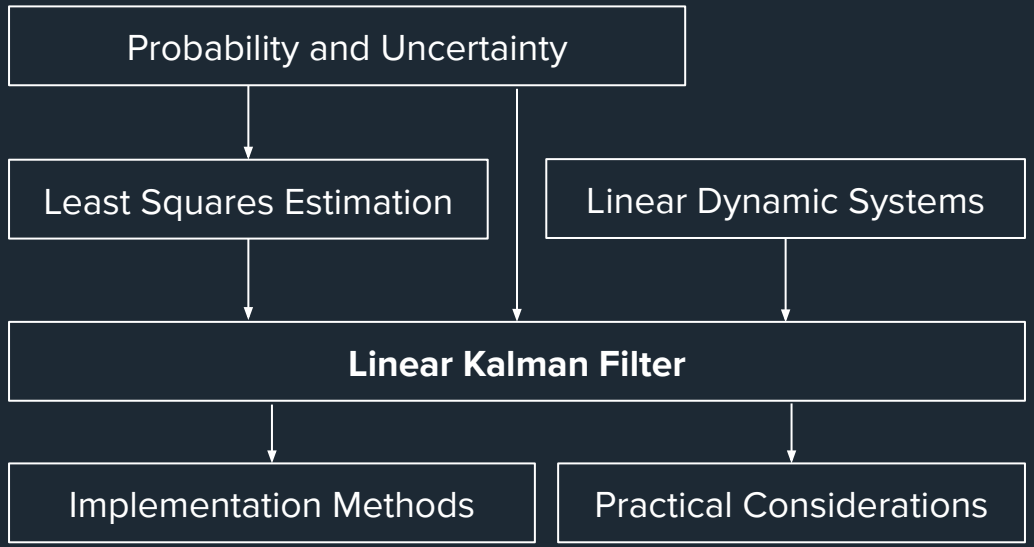
- Represent the **estimated state** as a **probability distribution**.
- Represent the **measurements or observations** of the current state (or function of the states) as a **probability distribution**.
- Fuse the two distributions to get a better estimate (**Bayes' Theorem**)
- **Prediction** process **increases** the uncertainty with time.
- **Update**/Measurement process **decreases** the uncertainty.
- Kalman Filter allows use to do this **numerically** and **mathematically** simply, by making a few **assumptions** about the probability distributions and a few other properties of the dynamic system.

Steven Dumble, PhD

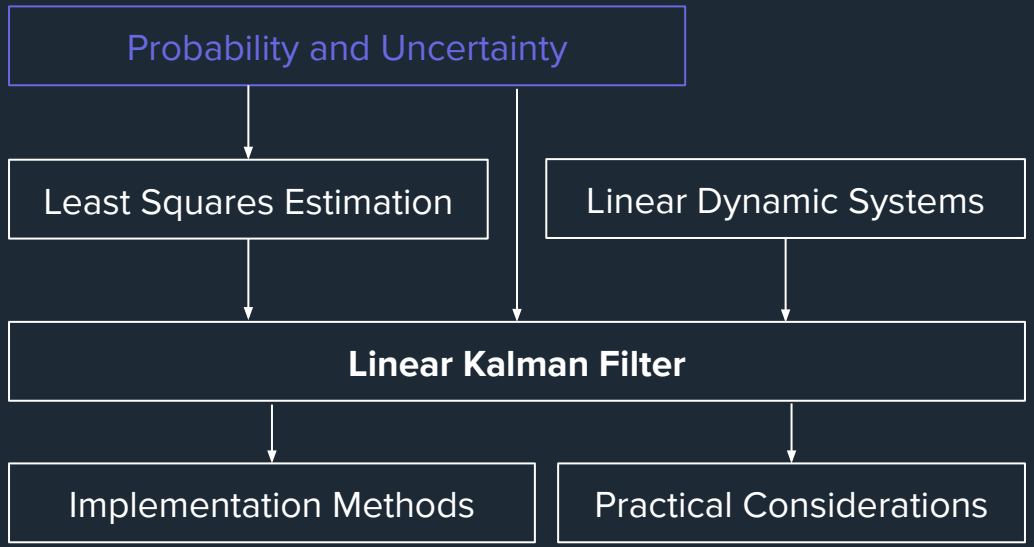
Introduction:
Learning Roadmap

Steven Dumble, PhD

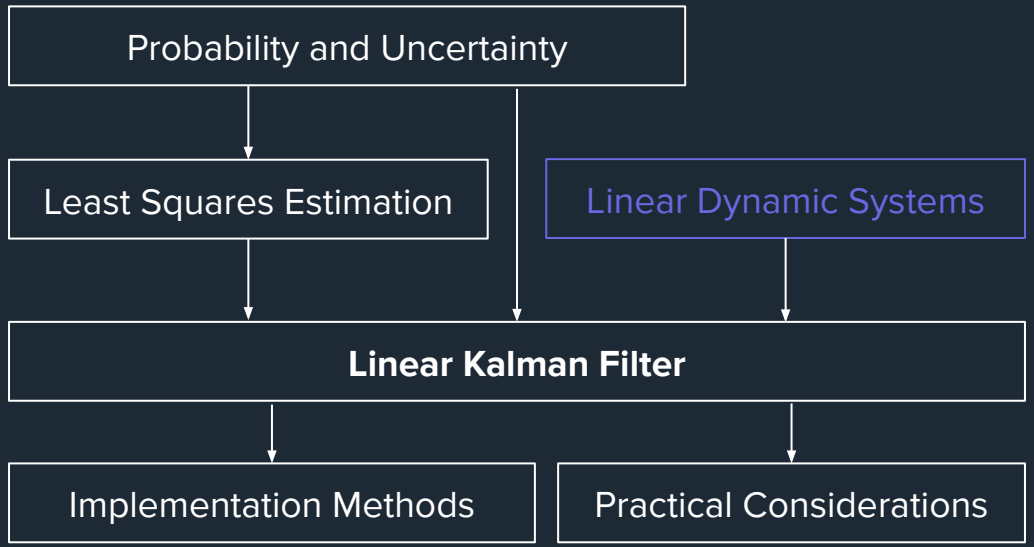
Learning Roadmap



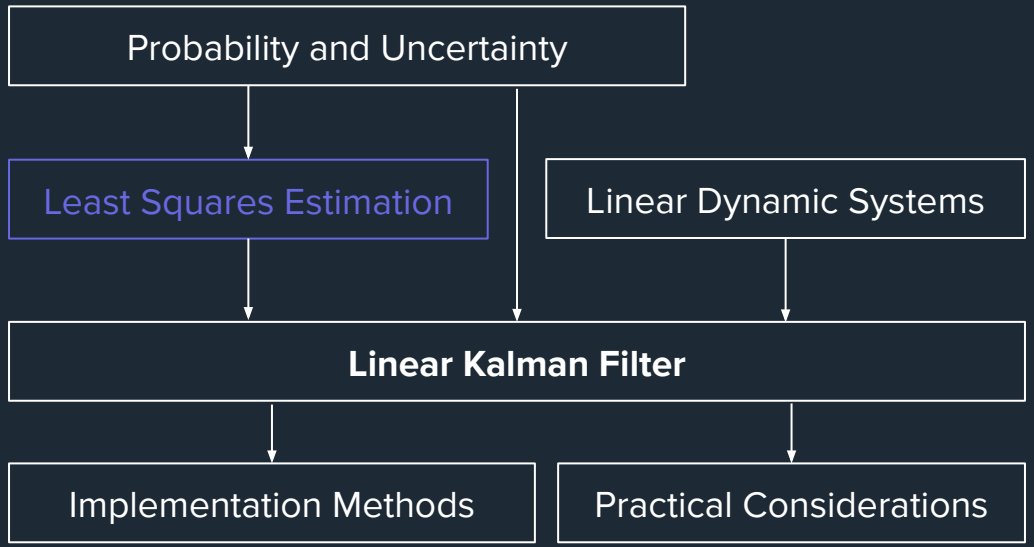
Learning Roadmap



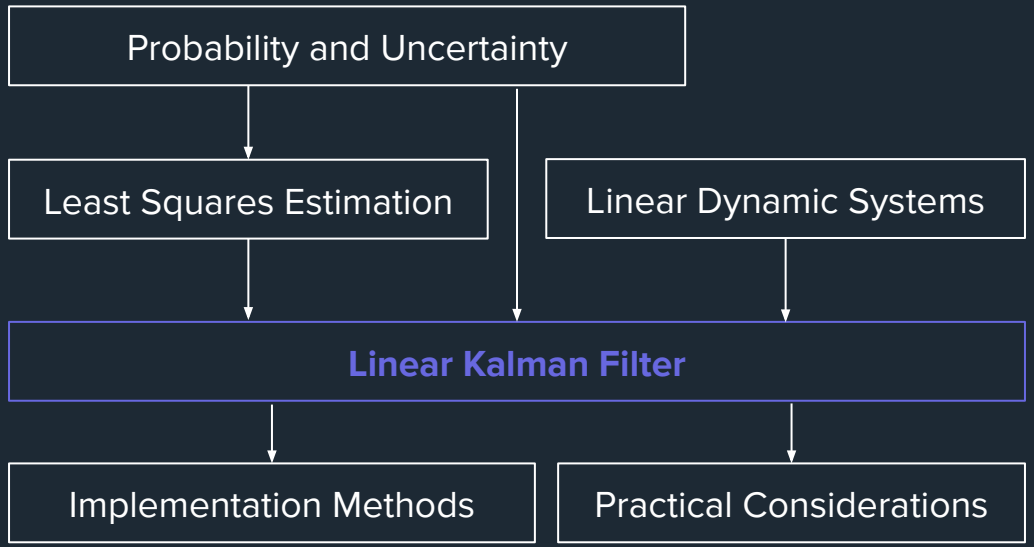
Learning Roadmap



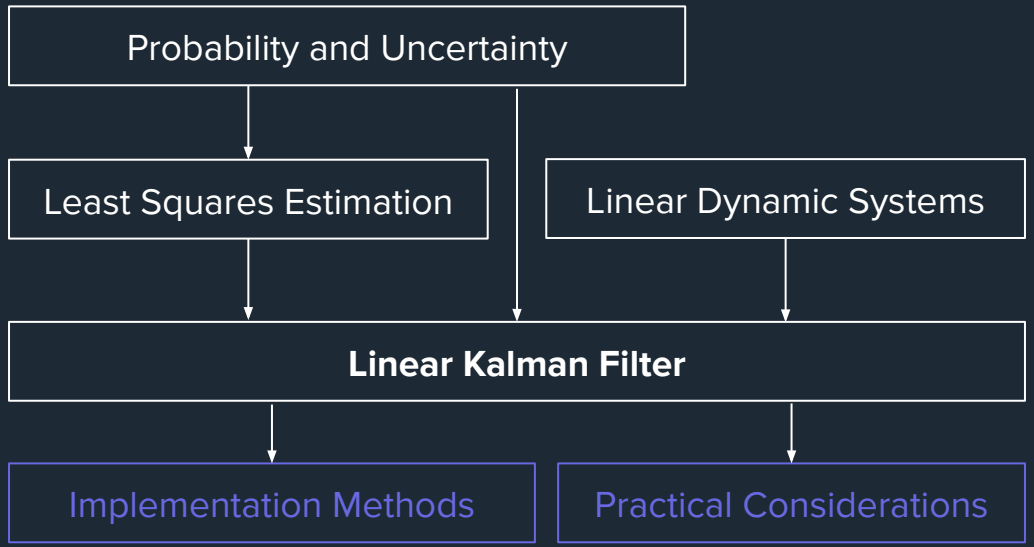
Learning Roadmap



Learning Roadmap



Learning Roadmap



Steven Dumble, PhD

Technitute_

Data Fusion with the Linear Kalman Filter

Introduction Series

Probability

Steven Dumble, PhD

Technitute_

Probability:
Basic Probability

Steven Dumble, PhD

Probability is a mathematical way of described the likelihood of an event happening.

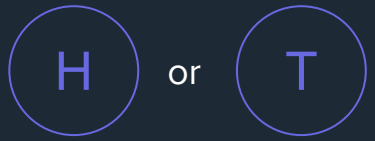
For any Event A, the probability is:

$$0 \leq P(A) \leq 1$$

P(A) = 0: Event A will never occur
P(A) = 1: Absolute certainty that Event A will occur

Basic Probability

Single Coin Toss:



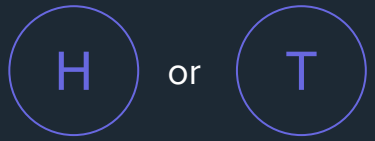
$$S = \{H, T\}$$
$$P(heads) = 0.5$$
$$P(tails) = 0.5$$

10x Coin Toss:

$$\#Heads \approx 5$$
$$\#Tails \approx 5$$

Basic Probability

Single Coin Toss Events:



$$S = \{H, T\}$$

$$P(H) = 0.5 \qquad P(T) = 0.5$$

$$P(H) + P(T) = 1$$

Basic Probability

Sum of all probabilities of each event or outcome in set S must equal 1:

$$\sum_{e \in S} P(e) = 1$$

Probability of event A occurring, assuming each event is equally as likely:

$$P(A) = \frac{Number\ of\ ways\ Event\ A\ occurs}{Total\ number\ of\ outcomes}$$

Basic Probability

Double Coin Toss Events: $S = \{HH, HT, TH, TT\}$

$P(HH) = \frac{1}{4}$

$P(A) = \frac{N(A)}{N(S)}$

$P(HT) = \frac{1}{4}$

$P(TH) = \frac{1}{4}$

$P(TT) = \frac{1}{4}$

Basic Probability

$$P(\neg A) = 1 - P(A)$$

Single Dice Roll Events: $S = \{1, 2, 3, 4, 5, 6\}$

Probability of Rolling a 2:

$$P(2) = \frac{1}{6}$$

Probability of Rolling any number:

$$P(S) = 1$$

Probability of Rolling any number other than 2:

$$P(\neg 2) = 1 - P(2) = 1 - 1/6 = 5/6$$

Basic Probability

Total Probability:

$$\sum_{e \in S} P(e) = 1$$

Probability of event A occurring, assuming each event is equally as likely:

$$P(A) = \frac{Number\ of\ ways\ Event\ A\ occurs}{Total\ number\ of\ outcomes}$$

Probability of Event A Not Occuring:

$$P(\neg A) = 1 - P(A)$$

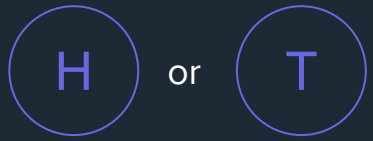
Technitute_

Probability:
Mutual Exclusivity

Steven Dumble, PhD

Mutual Exclusivity

Single Coin Toss Events: $S = \{H, T\}$



Single Dice Roll Events: $S = \{1, 2, 3, 4, 5, 6\}$

Mutual Exclusivity

Given a set of possible events A and B:

$$S = \{A, B\}$$

Basic probability says:

$$P(A \ or \ B) = P(A) + P(B) = 1$$

If events A and B occurring simultaneously is impossible:

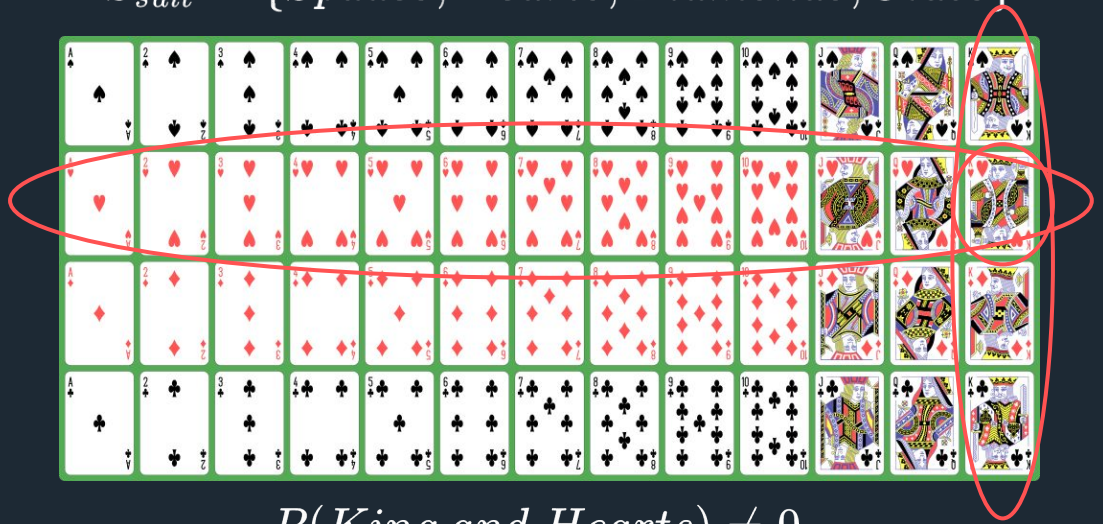
$$P(A \ and \ B) = 0$$

Then Event A and Event B are ***mutually exclusive.***

Non-Mutual Exclusivity

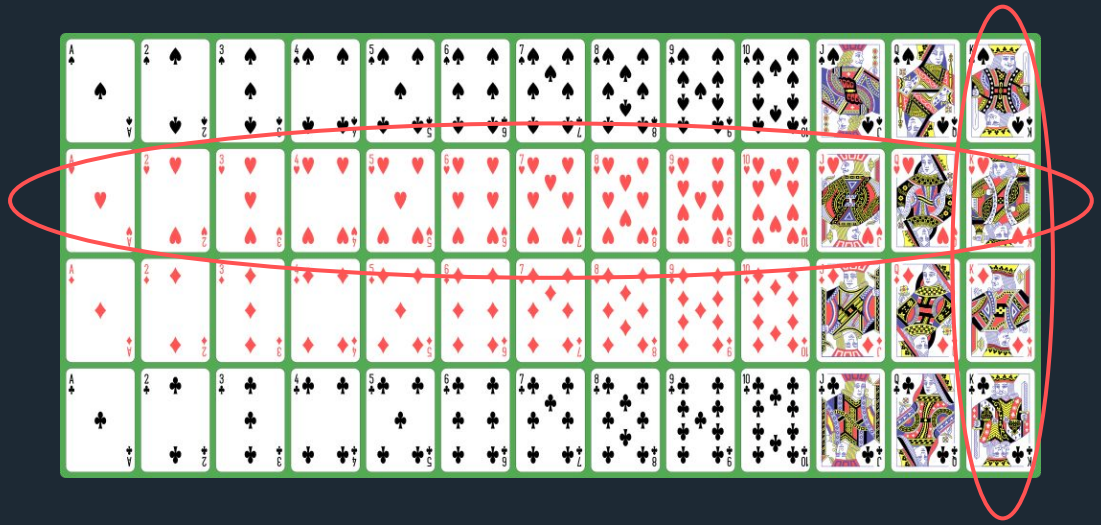
Events do not have to be mutually exclusive, it is possible to get different events occuring at the same time.

$$S_{number} = \{A, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K\}$$
$$S_{suit} = \{Spades, Hearts, Diamonds, Clubs\}$$



$$P(King\ and\ Hearts) \neq 0$$

Non-Mutual Exclusivity



$P(Hearts) = 13/52$

$P(King) = 4/52$

$P(King\ or\ Heart) \neq P(King) + P(Heart)$

$P(King\ or\ Heart) = P(King) + P(Heart) - P(King\ and\ Heart)$

$P(King\ or\ Heart) = 4/52 + 13/52 - 1/52 = 16/52$

Mutual Exclusivity

Mutually Exclusive:

$$P(A \text{ and } B) = 0$$

Non-Mutually Exclusive:

$$P(A \text{ and } B) \neq 0$$

Probability of Non Mutually Exclusive Events

$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$$

Technitute_

Probability:
Conditional Probability

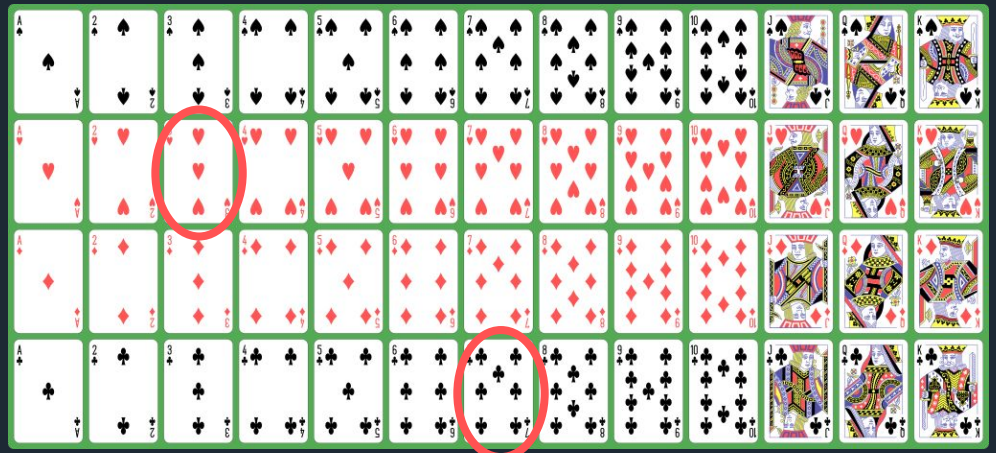
Steven Dumble, PhD

Conditional Probability

- Events can be considered ***independent*** if the likelihood of one event does not affect the likelihood of another occuring. (I.e Roll of a dice, toss of a coin)
- ***Dependant*** events are the opposite. When one event occurs it changes the probability of the other events.

Conditional Probability

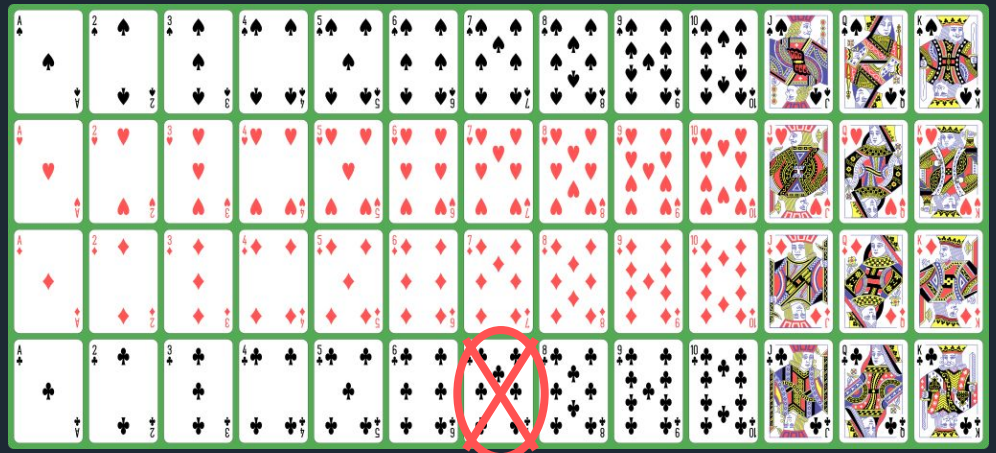
- Pick any card from a **full deck** of cards at random, there is a 50/50 chance that it is a black or red card.



$$P(red) = 26/52 = 0.5$$
$$P(black) = 26/52 = 0.5$$

Conditional Probability

- Pick any card at random again, without the last card replaced, then the probability changes.



$$P(red) = \frac{26}{52-1} = 0.51$$

$$P(black) = \frac{26-1}{52-1} = 0.49$$

Conditional Probability

The probability of Event A and Event B occurring if they are dependent:

$$P(A \text{ and } B) = P(A)P(B|A)$$

The probability of drawing a red card followed by a black card, without replacing the cards after they are drawn are:

$$P(red \text{ and } black) = P(red)P(black|red)$$
$$P(red \text{ and } black) = (26/52)(26/51) = 0.255$$

Conditional Probability

Conditional Probability:

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)}$$

Conditional Probability

Joint Probability

Marginal Probability

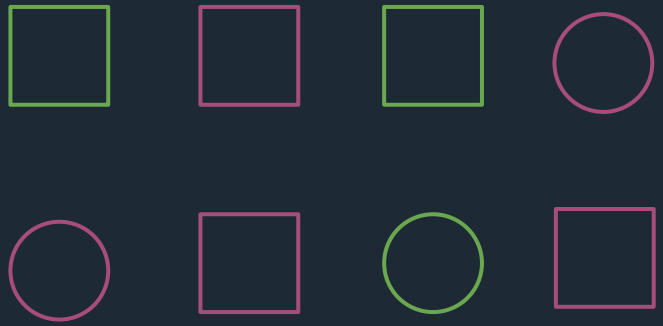
Independent Events:

$$P(A \text{ and } B) = P(A)P(B)$$

$$P(A|B) = P(A)$$

$$P(B|A) = P(B)$$

Conditional Probability



$$P(circle) = 3/8$$
$$P(square) = 5/8$$
$$P(circle\ and\ green) = 1/8$$

$$P(green \mid circle) = \frac{P(green\ and\ circle)}{P(circle)}$$

$$P(green \mid circle) = \frac{1/8}{3/8} = 1/3$$

Steven Dumble, PhD

Technitute_

Probability: Bayes' Theorem

Steven Dumble, PhD

Bayes' Theorem

- Bayes' Theorem or Bayes' Rule is one of the most important concepts used in Bayesian Estimation (i.e. Probabilistic Estimation)
- Bayes' Theorem allows you to calculate the likelihood or bounds on an unknown parameter or event based on prior information related to that event (Bayesian inference)

Bayes' Theorem

From conditional probability we know:

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)} \qquad P(B|A) = \frac{P(B \text{ and } A)}{P(A)}$$

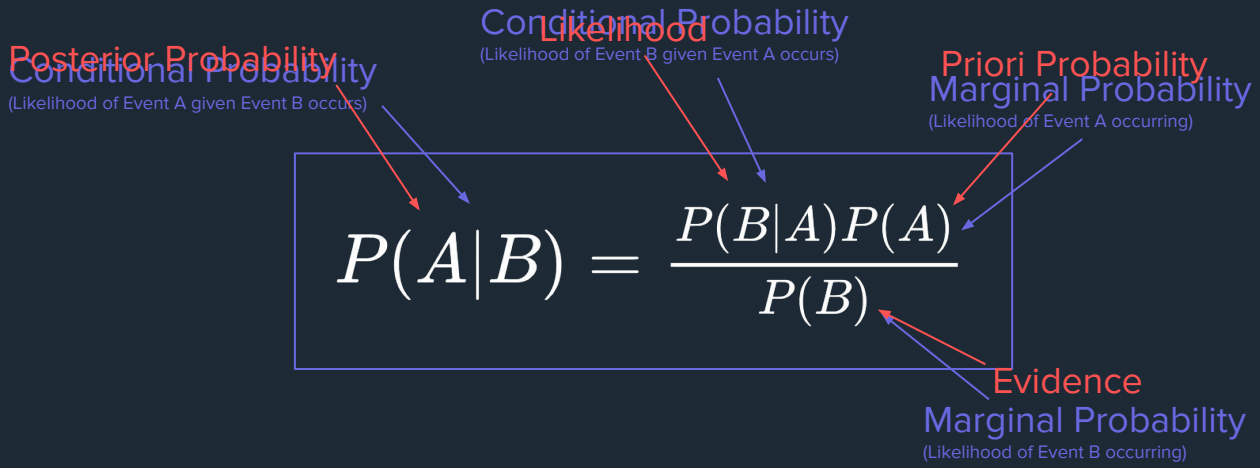
We also know that the joint probabilities are equivalent:

$$P(A \text{ and } B) = P(B \text{ and } A)$$

Substituting the condition probability equations into each other gives Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes' Theorem



Bayes' Theorem Example

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

What is the likelihood of rain today?

Prior Information: $P(rain) = 0.6$ (i.e. Its rained 18 days out of the last 30 days)

Evidence: $P(cloudy) = 0.48$ (i.e. You look outside and say there is a 48% chance of it being cloudy this morning)

Likelihood: $P(cloudy|rain) = 0.68$ (i.e. On the days that it has rained, 68% of the time it was cloudy in the morning)

Posterior Probability: $$P(rain|cloudy) = \frac{P(cloudy|rain)P(rain)}{P(cloudy)}$$

$$P(rain|cloudy) = 0.85$$

Better take an umbrella!

Steven Dumble, PhD

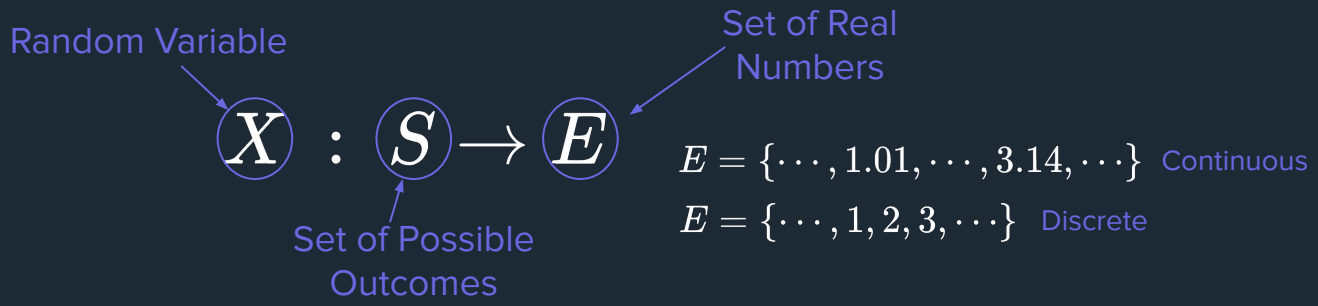
Technitute_

Probability:
Random Variables

Steven Dumble, PhD

Random Variable

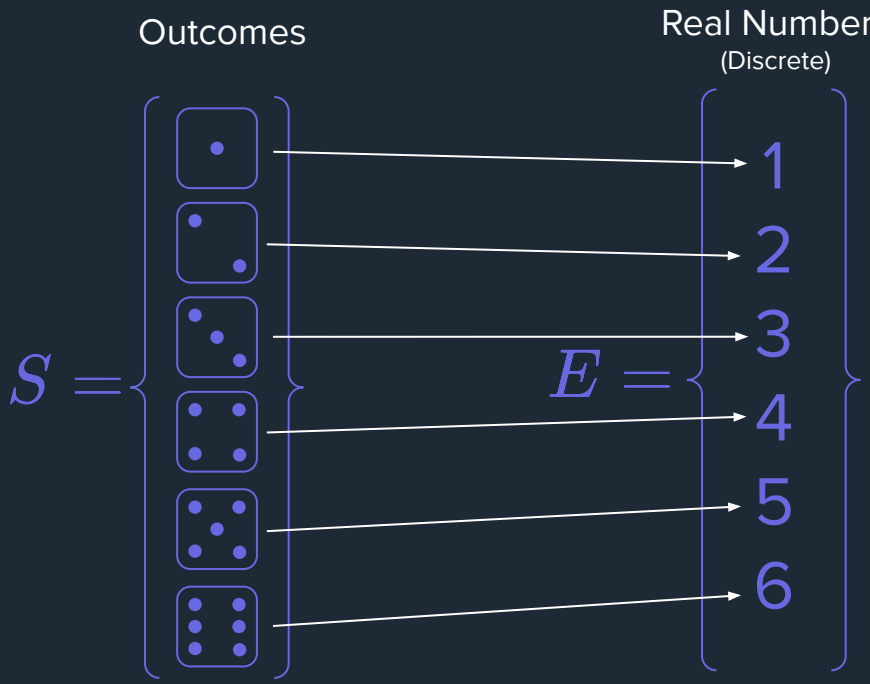
- A random variable is a way to mathematically express stochastic outcomes as real numbers.
- Functional mapping that maps a set of experimental outcomes to a set of real numbers.



Random Variable

$$X : S \to E$$

Let X be the random variable for the toss of a dice



The random variable X has the the possibility of being any of the real numbers in E.

5

However once the experiment (i.e. toss of dice) has been carried out the value of the random variable X or outcome has been determined. (I.e. it could 3 or it could be 5, etc)

You can treat the random value as any more algebraic value.

Steven Dumble, PhD

Technitute_

Probability:
Probability Density Function

Steven Dumble, PhD

Probability Density Function

- Let's say we have a random variable X, but we would like to know how likely each outcome is to occur. Is each outcome equally as likely (ie. a toss of a coin) or are some outcomes more likely to occur than others?
- How can we describe and quantify this?
- This is done with ***Probability Density Functions***.

Probability Density Functions

- Probability Density Function (PDF) - Continuous Random Variables
- Probability Mass Function (PMF) - Discrete Random Variables

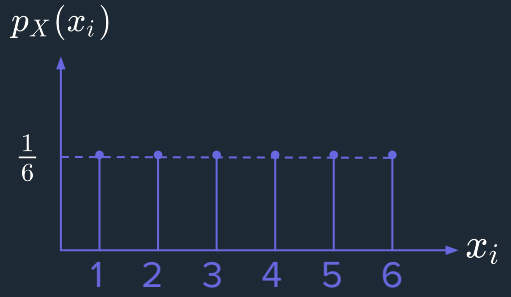
Measure the relative likelihood or probability that a specific outcome will occur.

Dice Probability Mass Function

$p_X(x_i)$

Outcome of a Dice Roll - Discrete (PMF)

$$P(X=1)=\frac{1}{6}$$
$$P(X=2)=\frac{1}{6}$$
$$\vdots$$
$$P(X=6)=\frac{1}{6}$$



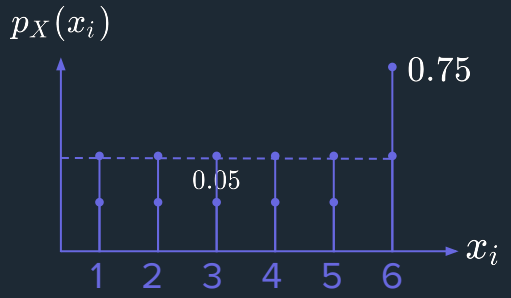
$$p_X(x_i)=P(X=x_i)$$
$$p_X(x_i)=\frac{1}{6}$$

Dice Probability Mass Function

$p_X(x_i)$

Outcome of a Weighted Dice Roll - Discrete (PMF)

$$P(6) = 0.75$$
$$P(1) = 0.05$$
$$\vdots$$
$$P(5) = 0.05$$



$$\sum P(X = x_i) = 1$$

$$P(1 \leq X \leq 5) = \sum_{x_i=1}^{5} P(X = x_i) = 0.25$$

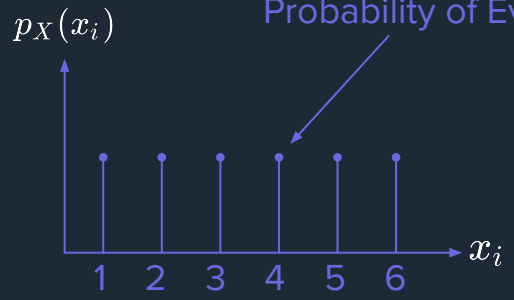
$$P(\neg 6) = 1 - P(X = 6) = 0.25$$

Probability Mass Function (PMF) (Discrete)

$$p_X(x_i)$$

$$p_X(x_i) = P(X = x_i)$$

Probability of Event



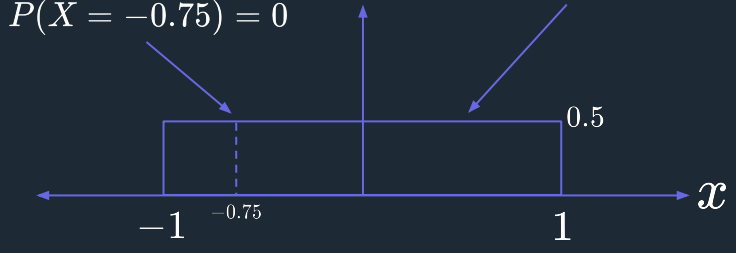
$$\sum P(X = x_i) = 1$$

Probability Density Function (PDF) (Continuous)

$$f_X(x)$$

$$P(a \leq X \leq b) = \int_a^b f_X(x)dx$$

Likelihood of Event



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

Probability Functions

- Probability Mass Function (PMF) - Discrete Random Variables $p_X(x_i)$

  $p_X(x_i) = P(X = x_i)$

  $\sum p_X(x_i) = 1$

- Probability Density Function (PDF) - Continuous Random Variables $f_X(x)$

  $P(a \leq X \leq b) = \int_a^b f_X(x)dx$ $\qquad$ $P(X = x) = 0$

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

Steven Dumble, PhD

Probability:
Expectation Operator

Steven Dumble, PhD

Expectation Operator

Let X be a discrete random variable and we will test the value N times. We observe the value A_1 occur n_1 times and the value A_2 occur n_2 times and so forth to value A_m occuring n_m times. The expected value $E(X)$ of the random variable X is computed with:

$$E(X) = \frac{1}{N} \sum_{i=1}^{m} A_i n_i$$

Expectation Operator

$$E(X) = \frac{1}{N} \sum_{i=1}^{m} A_i n_i$$

Dice Example $N = 20$

Die Value A_i	Occurrence n_i
1	2
2	4
3	2
4	3
5	5
6	4

$$E(X) = \frac{1}{20}[(1)(2) + (2)(4) + \cdots + (6)(4)] = 3.85$$

$$E(X) = \frac{1}{N}[(1)\frac{N}{6} + (2)\frac{N}{6} + \cdots + (6)\frac{N}{6}]$$

$$E(X) = 3.5$$

Expectation Operator

Discrete Random Variable

$$E(X) = \frac{1}{N} \sum_{i=1}^{m} A_i n_i$$

Continuous Random Variable

$$E(X) = \int_{-\infty}^{\infty} x f_X(x) dx$$

The expected value or mean of the random variable is usually for simplicity written as:

$$\bar{X} = \bar{x} = E(X)$$

Steven Dumble, PhD

Technitute_

Probability:
Distribution Statistical Properties

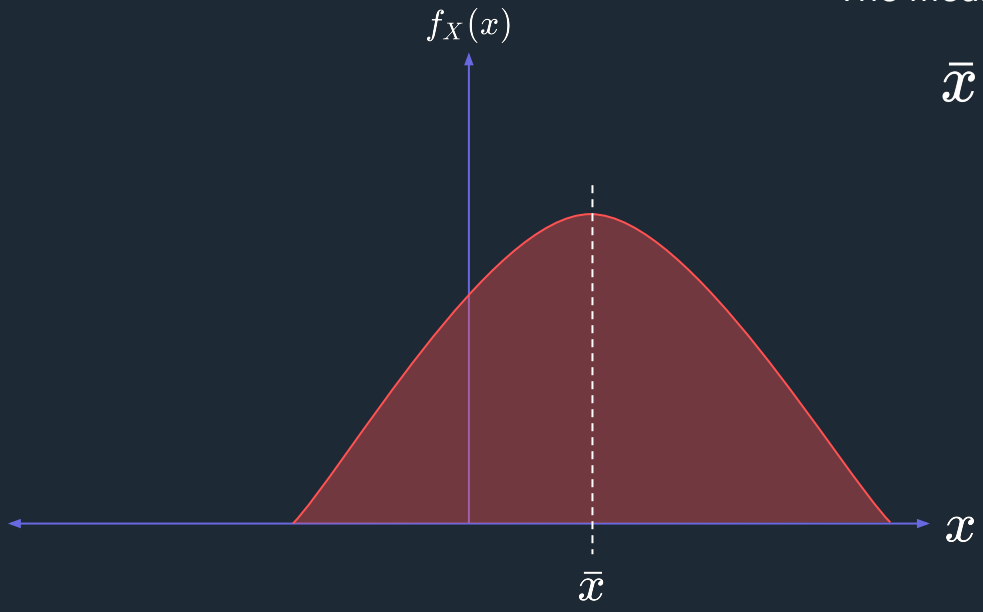
Steven Dumble, PhD

Distribution Statistical Properties

It is very useful to be able to describe a probability density function or distribution in a few key properties:

- Mean
- Variance
- Skew

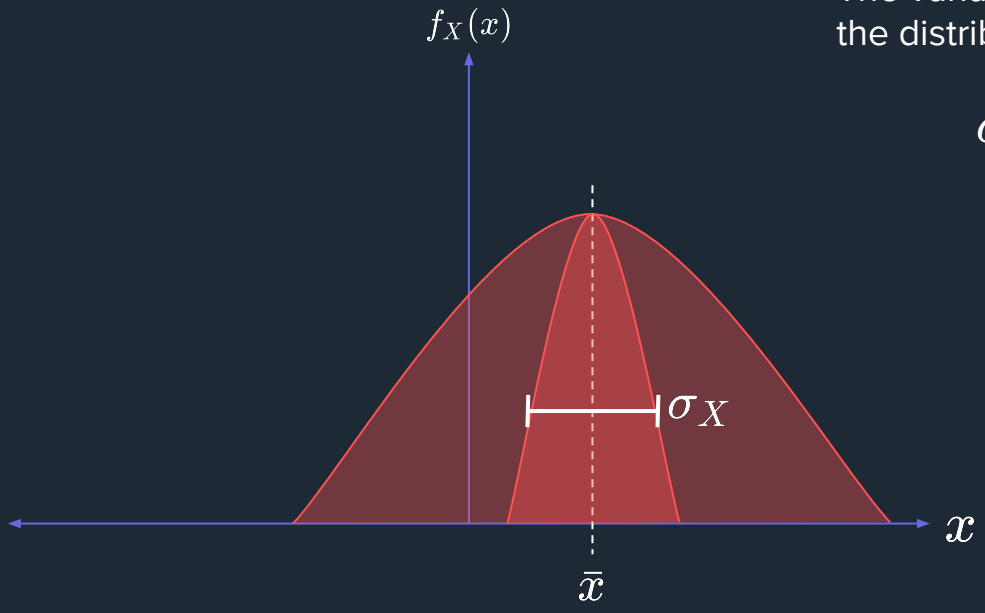
Distribution Mean



The mean of a distribution is just the expected value:

$$\bar{x} = E(X)$$

Distribution Variance



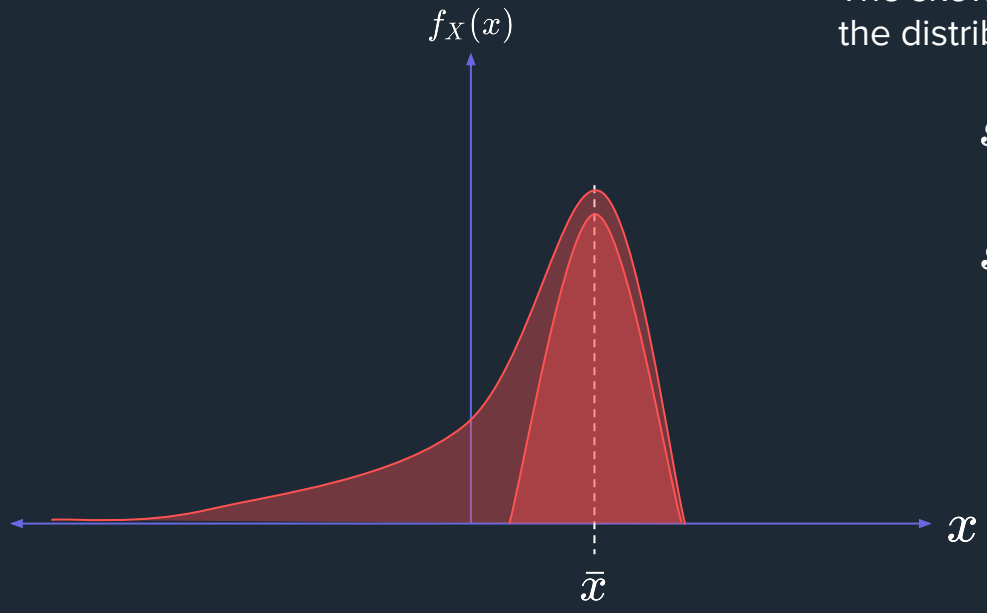
The variance of a distribution a measure of how much the distribution varies from the mean:

$$\sigma_X^2 = E\left[(X - \bar{x})^2\right]$$

$$\begin{aligned}\sigma_X^2 &= E\left[X^2 - X\bar{x} + \bar{x}^2\right] \\ &= E(X^2) - 2\bar{x}^2 + \bar{x}^2 \\ &= E(X^2) - \bar{x}^2\end{aligned}$$

$$X \sim (\bar{x}, \sigma_X^2)$$

Distribution Skewness



The skew of a distribution a measure the asymmetry of the distribution from the mean:

$$skew = E\left[(X - \bar{x})^3\right]$$

$$skewness = skew/\sigma_X^3$$

Distribution Statistical Properties

- Mean (First Moment)

$$\bar{x} = E(X)$$

- Variance (Second Moment)

$$\sigma_X^2 = E\left[(X - \bar{x})^2\right]$$

- Skew (Third Moment)

$$skew = E\left[(X - \bar{x})^3\right]$$

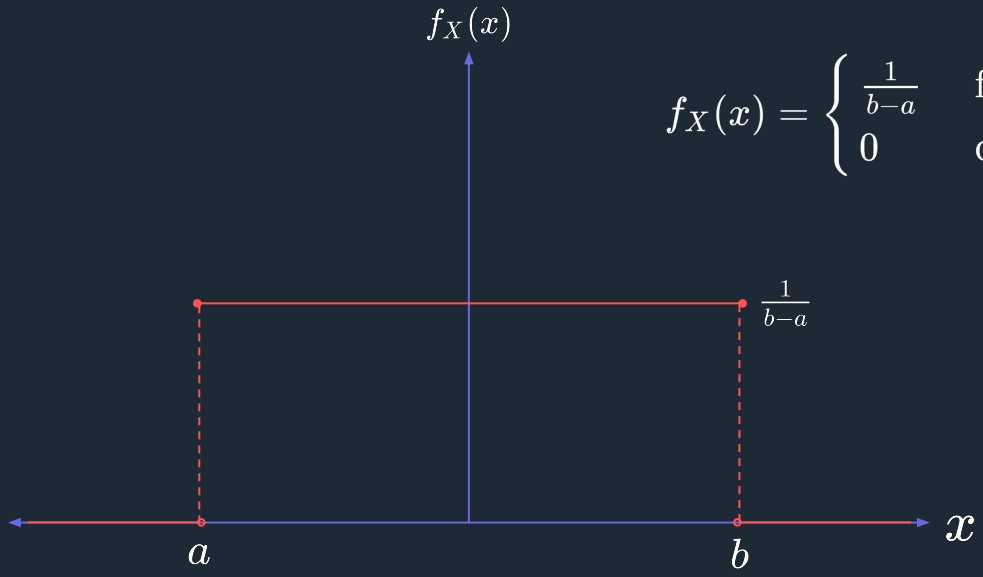
Steven Dumble, PhD

Probability:
Uniform Distribution

Steven Dumble, PhD

Uniform Distribution (Continuous)

$$X \sim U(a, b)$$



$$f_X(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

Mean: $\bar{x} = \frac{1}{2}(a + b)$

Variance: $\sigma_X^2 = \frac{1}{12}(b - a)^2$

Skew: $skewness = 0$

Steven Dumble, PhD

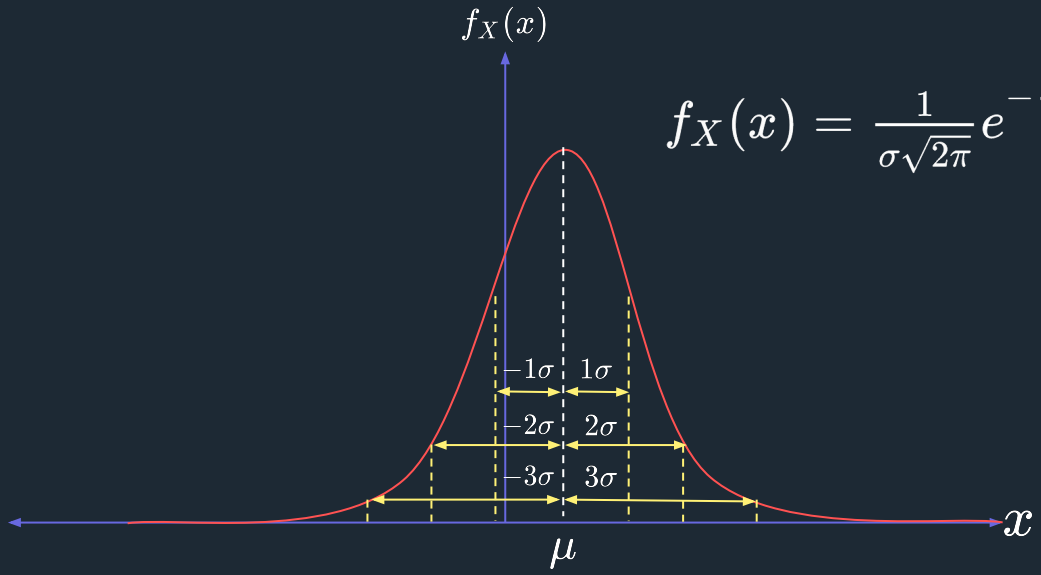
Technitute_

Probability:
Gaussian Distribution

Steven Dumble, PhD

Gaussian Distribution (Continuous)

$$X \sim N(\mu, \sigma^2)$$

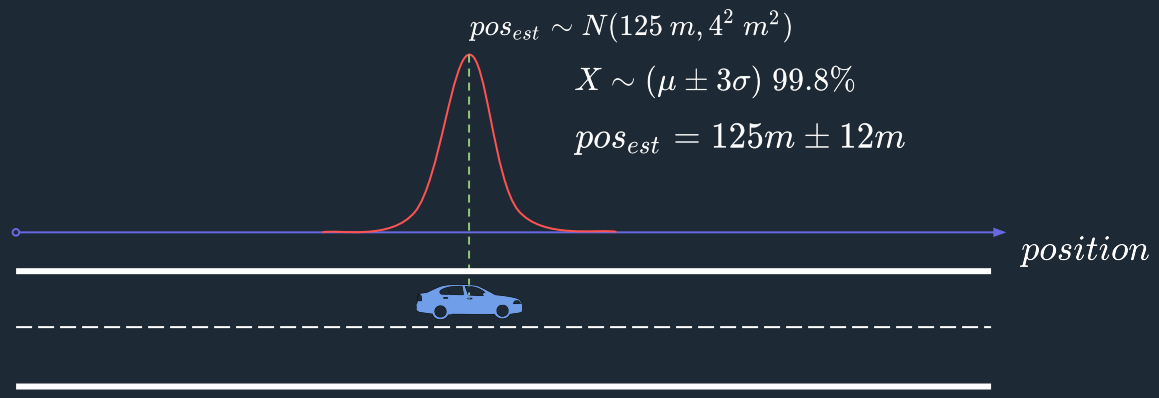


$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$$P(\mu - 1\sigma \leq X \leq \mu + 1\sigma) = 0.68$$
$$P(\mu - 2\sigma \leq X \leq \mu + 2\sigma) = 0.95$$
$$P(\mu - 3\sigma \leq X \leq \mu + 3\sigma) = 0.99$$

Gaussian Distribution (Continuous)

$$X \sim N(\mu, \sigma^2)$$



Steven Dumble, PhD

Technitute_

Probability: Gaussian Distribution Linear Transformation

Steven Dumble, PhD

Transformation of Random Variables

Random variables can be transformed through a function. Suppose we have a random variable X and its associated pdf $f_X(x)$, it is possible to apply a mathematical function $y = g(x)$ to the pdf and find the pdf $f_Y(y)$ of the transformed random variable Y.

Assume the transformation functions are monotonic

$$y = g(x)$$
$$x = g(y)^{-1} = h(y)$$

$$f_Y(y) = |h'(y)| \, f_X(h(y))$$

Linear Transformation of Gaussian PDF

Suppose we have the random variable:

$$X \sim N(\bar{x}, \sigma_x^2)$$

$$f_X(x) = \frac{1}{\sigma_x \sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\bar{x}}{\sigma_x}\right)^2\right]$$

Suppose we have the transformation:

$$Y = g(X) = aX + b$$

$$X = g^{-1}(Y) = \frac{Y-b}{a}$$

$$h(Y) = \frac{(Y-b)}{a}$$

$$h'(y) = \frac{1}{a}$$

Using the relationship:

$$f_Y(y) = |h'(y)| f_X(h(y))$$

We find:

$$\bar{y} = a\bar{x} + b$$

$$f_Y(y) = \frac{1}{a\sigma_x \sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{y-(a\bar{x}+b)}{a\sigma_x}\right)^2\right]$$

$$\sigma_y^2 = a^2 \sigma_x^2$$

Linear Transformation of Gaussian PDF

$$X \sim N(\bar{x}, \sigma_x^2) \longrightarrow Y = aX + b \longrightarrow Y \sim N(a\bar{x} + b, a^2\sigma_x^2)$$

A linear transformation of a Gaussian PDF is just another Gaussian PDF with the mean and variance transformed.

Steven Dumble, PhD

Technitute_

Probability:
Multiple Random Variables

Steven Dumble, PhD

Multiple Random Variables

Let X be a random variable with a pdf of $f_X(x)$ and also let Y be a random variable with a pdf of $f_Y(y)$. It is possible to define the pdf for the joint probability (i.e. probability of X and Y) as $f_{XY}(x,y)$ or $f(x,y)$ for short.

Joint Probability:

$$P(a \leq X \leq b \text{ and } c \leq Y \leq d) = \int_c^d \int_a^b f(x,y)\, dx\, dy$$

Marginal Density Functions

$$f_X(x) = \int_{-\infty}^{\infty} f(x, y)\, dy$$

$$f_Y(y) = \int_{-\infty}^{\infty} f(x, y)\, dx$$

Expected Value

Single Random Variable:

$$E[g(x)] = \int_{-\infty}^{\infty} g(x) f_X(x)\, dx$$

Multiple Random Variable:

$$E[g(x,y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x,y) f(x,y)\, dx\, dy$$

Independent Random Variables

Basic Probability Condition:

$$P(A \; and \; B) = P(A)P(B)$$

Density Function Condition:

$$f_{XY}(x, y) = f_X(x)f_Y(y)$$

Expected Value of Multiplication of Independent Random Variables

$$E(XY) = \int\int XY f(x,y)\, dx\, dy$$

$$f_{XY}(x,y) = f_X(x) f_Y(y)$$

$$E(XY) = \int\int XY f_X(x) f_Y(y)\, dx\, dy$$

$$E(XY) = \int X f_X(x)\, dx \int Y f_Y(y)\, dy$$

$$E(XY) = E(X)E(Y)$$

Assuming Independence

Expected Value of Sum of Independent Random Variables

$$z(x, y) = g(x) + h(y)$$

$$E[z(x, y)] = E[g(x) + h(y)]$$

$$E(z) = \int\int [g(x) + h(x)] f(x, y)\, dx\, dy$$

$$E(z) = \int\int g(x) f_X(x) f_Y(y)\, dx\, dy + \int\int h(y) f_X(x) f_Y(y)\, dx\, dy$$

$$E(z) = \int g(x) f_X(x)\, dx \int f_Y(y)\, dy + \int h(y) f_Y(y)\, dy \int f_X(x)\, dx$$

1

1

$$E(z) = E[g(x)] + E[h(y)]$$

$$z = x + y$$

$$E(x + y) = E(x) + E(y)$$

Dependant Random Variables

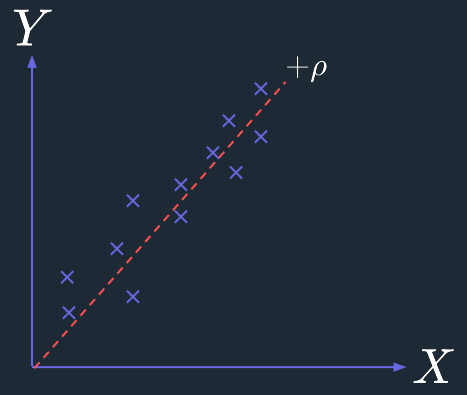
The random variables might not be independent, there might be some correlation between the two.

Covariance:

$$\begin{aligned} C_{XY} &= E[(X - \bar{x})(Y - \bar{y})] \\ &= E(XY) - \bar{x}\bar{y} \end{aligned}$$

Correlation Coefficient:

$$\rho = \frac{C_{XY}}{\sigma_x \sigma_y}$$



Multiple Random Variables

Joint Probability:

$$P(a \leq X \leq b \text{ and } c \leq Y \leq d) = \int_c^d \int_a^b f(x,y)\, dx\, dy$$

Expected Value:

$$E[g(x,y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x,y) f(x,y)\, dx\, dy$$

Independence Condition:

$$f_{XY}(x,y) = f_X(x) f_Y(y)$$

Covariance:

$$C_{XY} = E[(X - \bar{x})(Y - \bar{y})]$$

Marginal Density Functions:

$$f_X(x) = \int_{-\infty}^{\infty} f(x,y)\, dy$$

$$f_Y(y) = \int_{-\infty}^{\infty} f(x,y)\, dx$$

Independent Expectation Identities:

$$E(XY) = E(X)E(Y)$$

$$E(X+Y) = E(X) + E(Y)$$

Distribution Statistical Properties

- Mean (First Moment)

$$\bar{x} = E(X)$$

- Variance (Second Moment)

$$\sigma_X^2 = E\left[(X - \bar{x})^2\right]$$

- Skew (Third Moment)

$$skew = E\left[(X - \bar{x})^3\right]$$

Steven Dumble, PhD

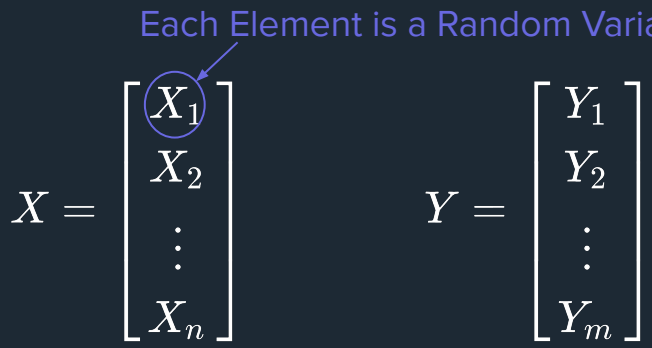
Technitute_

Probability: Multivariate Statistics

Steven Dumble, PhD

Multivariate Random Vectors

We can generalize single random variables into vector form:



Multivariate Mean

The mean can be calculated in the same way, but on a per element basis:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \qquad \bar{X} = E(X) = \begin{bmatrix} E(X_1) \\ E(X_2) \\ \vdots \\ E(X_n) \end{bmatrix}$$

Multivariate Covariance

The covariance between two random variable vectors can be calculated, which now forms a matrix of covariance values:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$$

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{bmatrix}$$

$$\begin{aligned} C_{XY} &= E[(X - \bar{X})(Y - \bar{Y})^T] \\ &= E(XY^T) - \bar{X}\bar{Y}^T \end{aligned}$$

$$C_{XY} = \begin{bmatrix} C_{X_1Y_1} & C_{X_1Y_2} & \cdots & C_{X_1Y_m} \\ C_{X_2Y_1} & C_{X_2Y_2} & \cdots & C_{X_2Y_m} \\ \vdots & \vdots & \ddots & \vdots \\ C_{X_nY_1} & C_{X_nY_2} & \cdots & C_{X_nY_m} \end{bmatrix}$$

Autocorrelation Matrix

The autocorrelation matrix, is simply the covariance matrix for the random vector and itself:

$$C_X = E[(X - \bar{X})(X - \bar{X})^T]$$

Variances

Cross-Covariances

$$C_X = \begin{bmatrix} \sigma^2_{X_1} & \sigma_{X_1 X_2} & \cdots & \sigma_{X_1 X_n} \\ \sigma_{X_2 X_1} & \sigma^2_{X_2} & \cdots & \sigma_{X_2 X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{X_n X_1} & \sigma_{X_n X_2} & \cdots & \sigma^2_{X_n} \end{bmatrix}$$

Covariance Matrix Properties

$$C_X = \begin{bmatrix} \sigma^2_{X_1} & \sigma_{X_1X_2} & \cdots & \sigma_{X_1X_n} \\ \sigma_{X_2X_1} & \sigma^2_{X_2} & \cdots & \sigma_{X_2X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{X_nX_1} & \sigma_{X_nX_2} & \cdots & \sigma^2_{X_n} \end{bmatrix}$$

The covariance matrix (autocorrelation) is:

- Symmetric
- Positive Semidefinite

$$\sigma_{ij} = \sigma_{ji}$$

$$C_X = C_X^T$$

$$z^T C_X z \geq 0$$

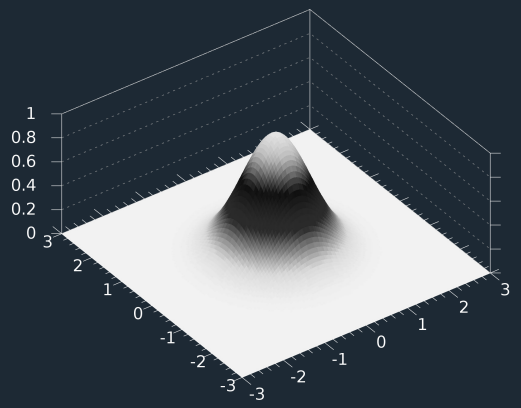
Steven Dumble, PhD

Technitute_

Probability: Multivariate Gaussian Distribution

Steven Dumble, PhD

Multivariate Gaussian Distribution



Random Variable:

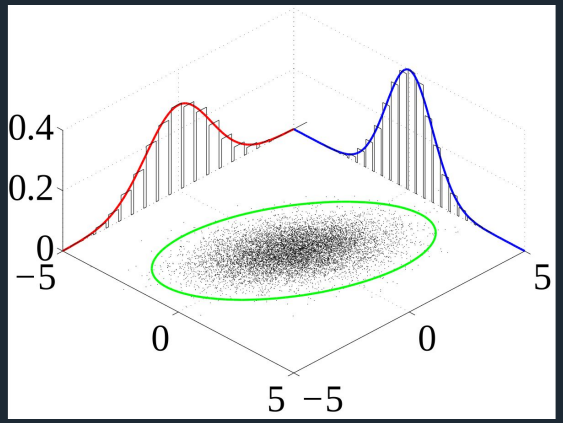
$$X \sim N(\mu, \sigma^2) \qquad f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Random Vector:

$$X \sim N(\bar{X}, C_X) \qquad f_X(X) = \frac{1}{(2\pi)^{n/2}|C_X|^{1/2}} e^{\left[-\frac{1}{2}(X-\bar{X})^T C_X^{-1}(X-\bar{X})\right]}$$

Multivariate Gaussian Distribution

The mean shifts the centre of the distribution, the variance controls the spread in the different axes, while the cross-covariances control the orientation of the distribution.



Steven Dumble, PhD

Technitute_

Probability:
Linear Transform of Uncertainty

Steven Dumble, PhD

Linear Transformation of Gaussian Distribution

A linear transformation of a Gaussian PDF is just another Gaussian PDF with the mean and variance transformed.

$$X \sim N(\bar{x}, \sigma_x^2)$$

$$\downarrow$$

$$Y = aX + b$$

$$\downarrow$$

$$Y \sim N(a\bar{x} + b, a^2\sigma_x^2)$$

Linear Transformation of Multivariate Gaussian Distribution

Suppose we have the random vector:

$$X \sim N(\bar{X}, C_X)$$

Suppose we have the transformation:

$$Y = g(X) = AX + b$$

$$X = g^{-1}(Y) = A^{-1}Y - A^{-1}b$$

$$h(Y) = A^{-1}Y - A^{-1}b$$

$$h'(y) = A^{-1}$$

Using the relationship:

$$f_Y(y) = |h'(y)| \, f_X\left(h\left(y\right)\right)$$

We find:

$$f_Y(Y) = \frac{1}{(2\pi)^{n/2}|AC_XA^T|^{1/2}} e^{\left[-\frac{1}{2}(Y-\bar{Y})^T(AC_XA^T)^{-1}(Y-\bar{Y})\right]}$$

$$\bar{Y} = A\bar{X} + b \qquad C_Y = AC_XA^T$$

Linear Transformation of Multivariate Gaussian Distribution

A linear transformation of a Gaussian PDF is just another Gaussian PDF with the mean and variance transformed.

$$X \sim N(\bar{X}, C_X)$$

$$\downarrow$$

$$Y = AX + b$$

$$\downarrow$$

$$Y \sim N(A\bar{X} + b, AC_X A^T)$$

Linear Transformation of Uncertainty

If C_X represents the uncertainty covariance, then it can be transformed to another frame using the linear transform $y = Ax$ where the transformed covariance is given by $C_Y = AC_X A^T$.

Steven Dumble, PhD

Basic Probability

Total Probability:

$$\sum_{e \in S} P(e) = 1$$

Probability of event A occurring, assuming each event is equally as likely:

$$P(A) = \frac{Number\ of\ ways\ Event\ A\ occurs}{Total\ number\ of\ outcomes}$$

Probability of Event A Not Occuring:

$$P(\neg A) = 1 - P(A)$$

Mutual Exclusivity

Mutually Exclusive:

$$P(A \text{ and } B) = 0$$

Non-Mutually Exclusive:

$$P(A \text{ and } B) \neq 0$$

Probability of Non Mutually Exclusive Events

$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$$

Conditional Probability

Conditional Probability:

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)}$$

Conditional Probability

Joint Probability

Marginal Probability

Independent Events:

$$P(A \text{ and } B) = P(A)P(B)$$

$$P(A|B) = P(A)$$

$$P(B|A) = P(B)$$

Probability Functions

- Probability Mass Function (PMF) - Discrete Random Variables $p_X(x_i)$

  $p_X(x_i) = P(X = x_i)$

  $\sum p_X(x_i) = 1$

- Probability Density Function (PDF) - Continuous Random Variables $f_X(x)$

  $P(a \leq X \leq b) = \int_a^b f_X(x)dx$ $\qquad$ $P(X = x) = 0$

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

Expectation Operator

Discrete Random Variable

$$E(X) = \frac{1}{N} \sum_{i=1}^{m} A_i n_i$$

Continuous Random Variable

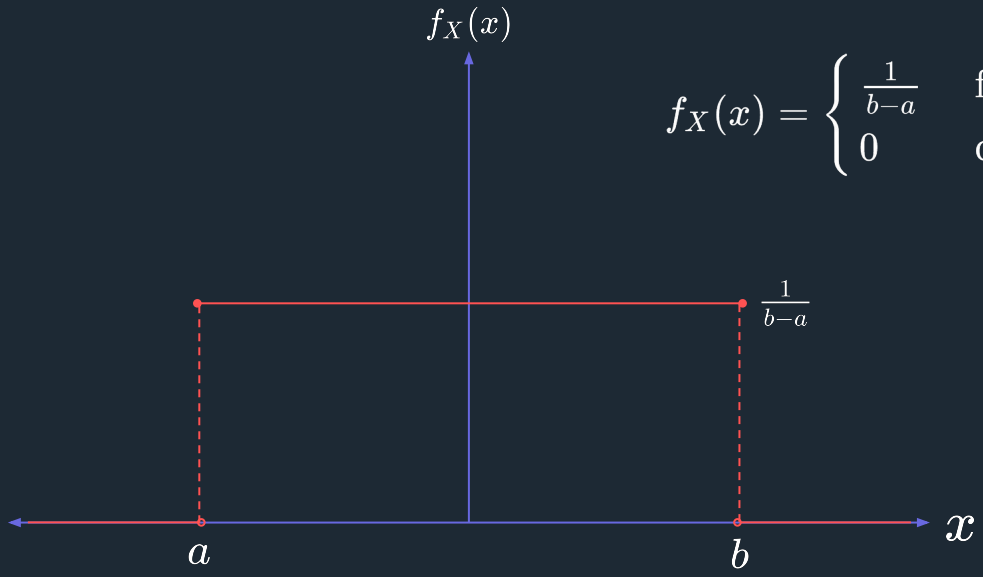
$$E(X) = \int_{-\infty}^{\infty} x f_X(x) dx$$

The expected value or mean of the random variable is usually for simplicity written as:

$$\bar{X} = \bar{x} = E(X)$$

Uniform Distribution (Continuous)

$$X \sim U(a, b)$$



$$f_X(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

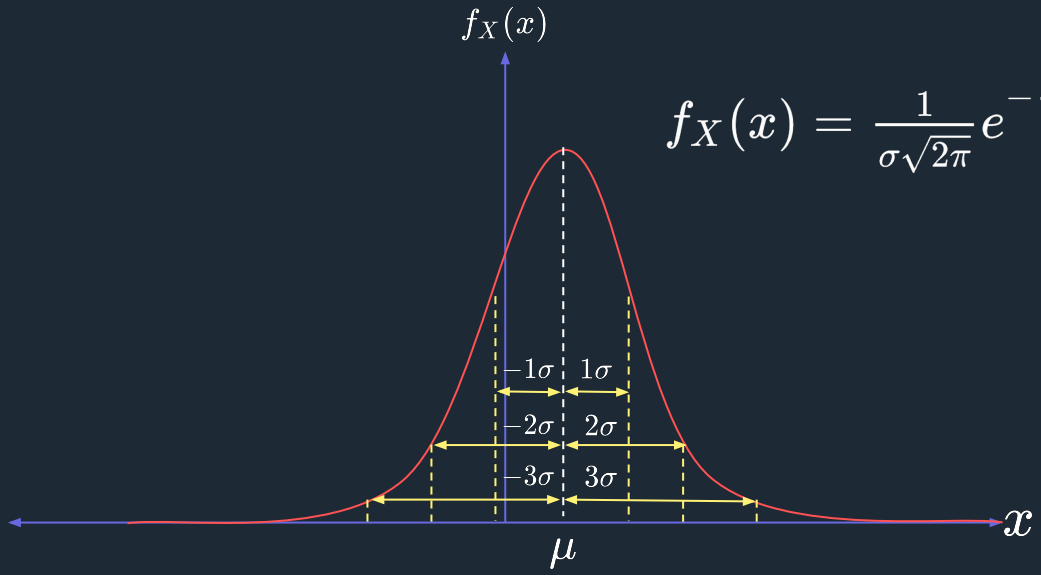
Mean: $\bar{x} = \frac{1}{2}(a + b)$

Variance: $\sigma_X^2 = \frac{1}{12}(b - a)^2$

Skew: $skewness = 0$

Gaussian Distribution (Continuous)

$$X \sim N(\mu, \sigma^2)$$



$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$$P(\mu - 1\sigma \leq X \leq \mu + 1\sigma) = 0.68$$
$$P(\mu - 2\sigma \leq X \leq \mu + 2\sigma) = 0.95$$
$$P(\mu - 3\sigma \leq X \leq \mu + 3\sigma) = 0.99$$

Multiple Random Variables

Joint Probability:

$$P(a \leq X \leq b \text{ and } c \leq Y \leq d) = \int_c^d \int_a^b f(x,y)\, dx\, dy$$

Expected Value:

$$E[g(x,y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x,y) f(x,y)\, dx\, dy$$

Independence Condition:

$$f_{XY}(x,y) = f_X(x) f_Y(y)$$

Covariance:

$$C_{XY} = E[(X - \bar{x})(Y - \bar{y})]$$

Marginal Density Functions:

$$f_X(x) = \int_{-\infty}^{\infty} f(x,y)\, dy$$

$$f_Y(y) = \int_{-\infty}^{\infty} f(x,y)\, dx$$

Independent Expectation Identities:

$$E(XY) = E(X)E(Y)$$

$$E(X+Y) = E(X) + E(Y)$$

Technitute_

Data Fusion with the Linear Kalman Filter

Introduction Series

Linear Systems

Steven Dumble, PhD

Technitute_

Linear Dynamic Systems: Differential Equations

Steven Dumble, PhD

Differential Equation:

An equation that relates one or more functions and their derivatives together.

Classical Mechanics:

Velocity and Position

$$v = \frac{dp}{dt}$$

Acceleration, Velocity and Position

$$a = \frac{dv}{dt} = \frac{d^2p}{dt^2}$$

Differential Equations

ODE

Ordinary Differential Equations

$$\frac{d}{dx} f(x)$$

$$a = \frac{dv}{dt} = \frac{d^2p}{dt^2}$$

PDE

Partial Differential Equations

$$\frac{\partial}{\partial x} f(x, y)$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Technitute_

Order of Differential Equation

n-th Equation Order:

$$f_n(\cdot)\frac{d^n y}{dx^n} + \cdots + f_1(\cdot)\frac{dy}{dx} + f_0(\cdot)y = g(\cdot)$$

Examples:

Second Order: $\frac{d^2 u}{dx^2} - x\frac{du}{dx} + u = 0$

First Order: $\frac{du}{dx} = a + u^2$

$$v = \frac{dp}{dt}$$

Velocity (First Order)

$$a = \frac{d^2 p}{dt^2}$$

Acceleration (Second Order)

Linear and Nonlinear

A **linear** system is a system which output changes **proportionally** with the input. Linear equations conform with the properties:

Additivity: $f(u+v) = f(u) + f(v)$

Homogeneity: $f(cu) = cf(u)$

Linear ODE

Both sides of the equation are a **linear combination** of the dependent variable and its derivatives.

$$a_n(x)\frac{d^n y}{dx^n} + \cdots + a_1(x)\frac{dy}{dx} + a_0(x)y = b(x)$$

Examples:

Linear ODE: $\frac{dy}{dx} + cy + x^2 = 0$

Non-Linear ODE: $\frac{dy}{dx} + a\sin(y) = 0$

Steven Dumble, PhD

Technitute_

Linear Dynamic Systems: State Space Representation

Steven Dumble, PhD

Dynamic Systems

- A system is a collection of interrelated entities (or differential equations) that can be considered as a whole.
- If the different process that make up this system change with time, then it is considered as a **dynamic system**.

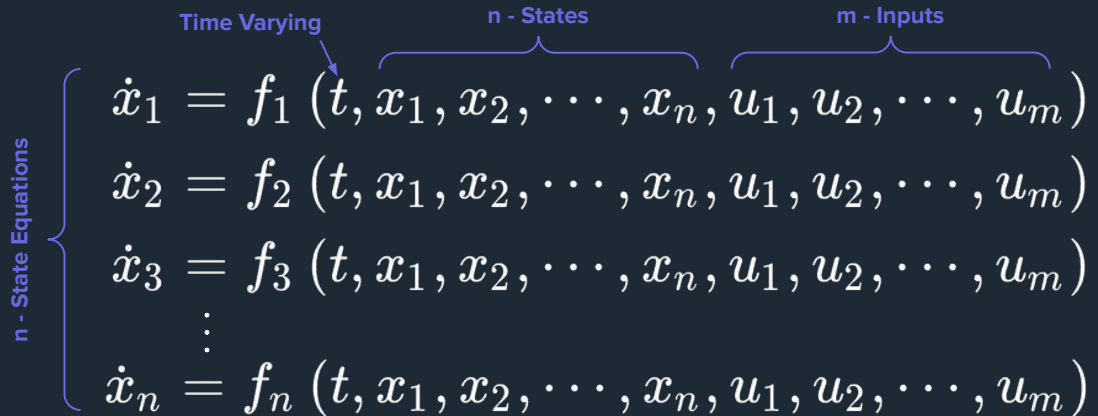
The differential equations that make up the system are the **state equations** of the dynamic system. The **state variables** of the system are the **dependant variables** of the state equations.

Dynamic Systems

$$\dot{x} = \frac{dx}{dt}$$

Newton's 'Dot' Notation
(derivative w/ time)

- Consider the system of time varying, first-order differential equations:



Dynamic Systems

This can compactly be written as:

$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$

where:

State Vector: $x(t) = \left[x_1(t), x_2(t), \cdots, x_n(t)\right]^T$

Input Vector: $u(t) = \left[u_1(t), u_2(t), \cdots, u_m(t)\right]^T$

Dynamic Systems

- Many of the processes that occur in the world can be expressed as linear or non-linear differential equations.
- We can describe these processes in state-space form which then allow us to use different **mathematical tools** to extract useful information and perform various analysis on the system.
- If the know the state of the system for the current time and all the current and future inputs to that system, then we can predict the values of the future states and outputs of that system and a lot more.

The Solar System is a Dynamic System

- Issac Newton's first use of **calculus** was to desibed the **differential equations** of the **orbit of objects** in the solar system.
- If you consider the solar system as a dynamic system, if you know the position of the planets for a given point in time, then you can predict where the planets will be at any point in the future or past.

Steven Dumble, PhD

Technitute_

Linear Dynamic Systems:
Differential Equations and State Space

Steven Dumble, PhD

Dynamic Systems and State Space

$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$

- First-Order Differential Equations make up the **state equations**
- The **state variables** of the system are the **dependant variables** of the state equations.
- How can we represent different ODEs in a state space form?
  - Newton's Equations of Motion: Force = Mass x Acceleration.

$$F(t) = m\ddot{x}(t)$$

  - How does the position x(t) evolve with time for a given force F(t)?

Dynamic Systems and State Space

2nd Order ODE

$$F(t) = m\ddot{x}(t) \quad \Rightarrow \quad \dot{x}(t) = f\left(t, x(t), u(t)\right)$$

$$\ddot{x}(t), \dot{x}(t), x(t)$$

$$a(t), v(t), x(t)$$

Acceleration, Velocity, Position

State Rates Vector: $\dot{x}(t) = \begin{bmatrix} \dot{v}(t) \\ \dot{x}(t) \end{bmatrix}$

State Vector: $x(t) = \begin{bmatrix} v(t) \\ x(t) \end{bmatrix}$

Input Vector: $u(t) = \begin{bmatrix} F(t) \end{bmatrix}$

$$\begin{bmatrix} \dot{v}(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{m}F(t) \\ v(t) \end{bmatrix}$$

Dynamic Systems and State Space

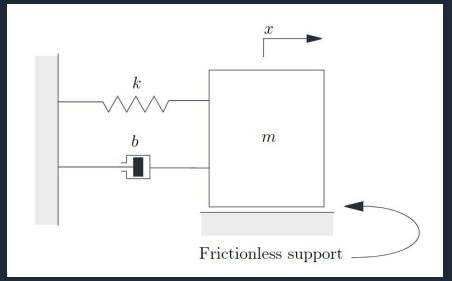
- You can turn a higher order ODE into first order state space representation.
- The states are all the lower order derivatives.

Steven Dumble, PhD

2nd Order Mechanical System Example

- Consider the 2nd Order Spring-Mass-Damper system:

$$\dot{v} = \frac{1}{m}(f - kx - bv)$$
$$\dot{x} = v$$



- Writing the system in state space form: $\dot{x}(t) = f\left(t, x(t), u(t)\right)$

State Rates Vector: $\dot{x}(t) = \begin{bmatrix} \dot{v}(t) \\ \dot{x}(t) \end{bmatrix}$

State Vector: $x(t) = \begin{bmatrix} v(t) \\ x(t) \end{bmatrix}$

Input Vector: $u(t) = \begin{bmatrix} f(t) \end{bmatrix}$

$$\begin{bmatrix} \dot{v} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(f - kx - bv) \\ v \end{bmatrix}$$

Technitute_

Linear Dynamic Systems:
Continuous and Discrete Time

Steven Dumble, PhD

Continuous Time

- The differential equations shown so far have all been in ‘continuous’ time.
  - Defined with respect to an independent variable (t) which varies continuously and smoothly
- For many practical purposes, we only really need to know the state of the system at a discrete set of point in time.

$$t_0, t_1, t_2, \cdots, t_{k-1}, t_k, t_{k+1}, \cdots$$

0.0s, 0.1s, 0.2s, ...

0.0s, 2.0s, 4.0s, ...

- This is what is called ‘discrete’ time.

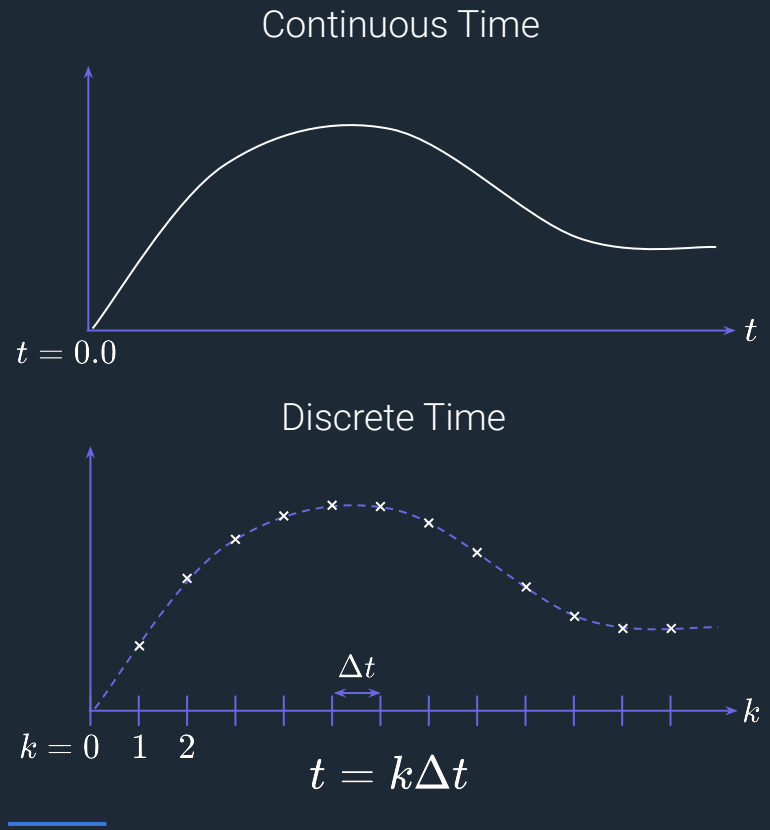
- Time steps forward in discrete blocks of time.
- Timestep/Sample Time is how large the blocks of time are.

$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$

$$x(t_{k+1}) = f\left(t_k, x(t_k), u(t_k)\right)$$

$$t_k = k\Delta t$$

(moves in constant steps, not rates)



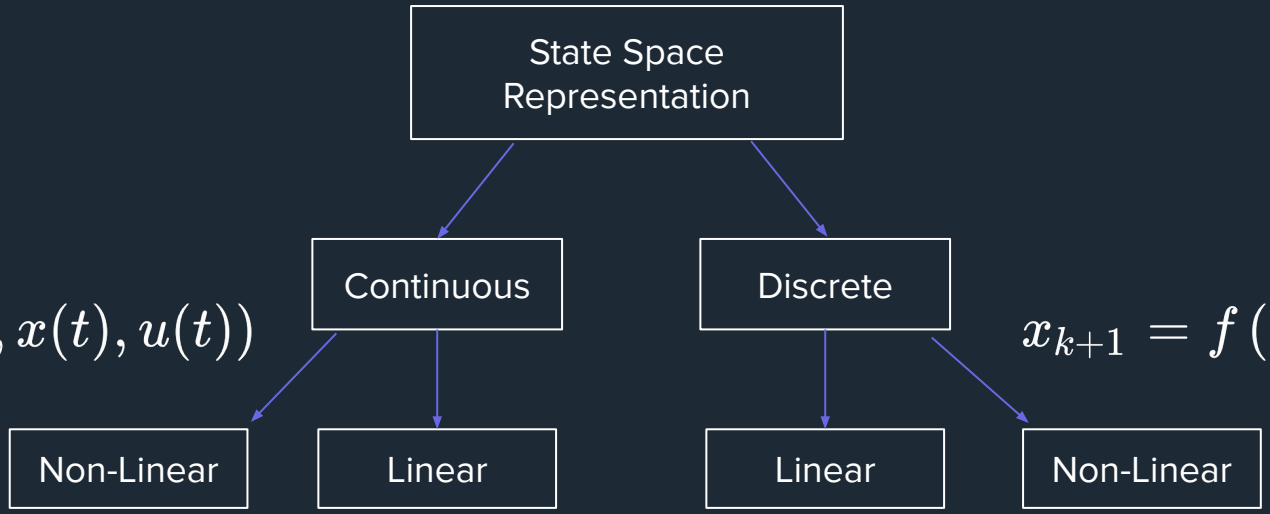
Steven Dumble, PhD

Technitute_

Linear Dynamic Systems:
Mathematical Models

Steven Dumble, PhD

Mathematical Models



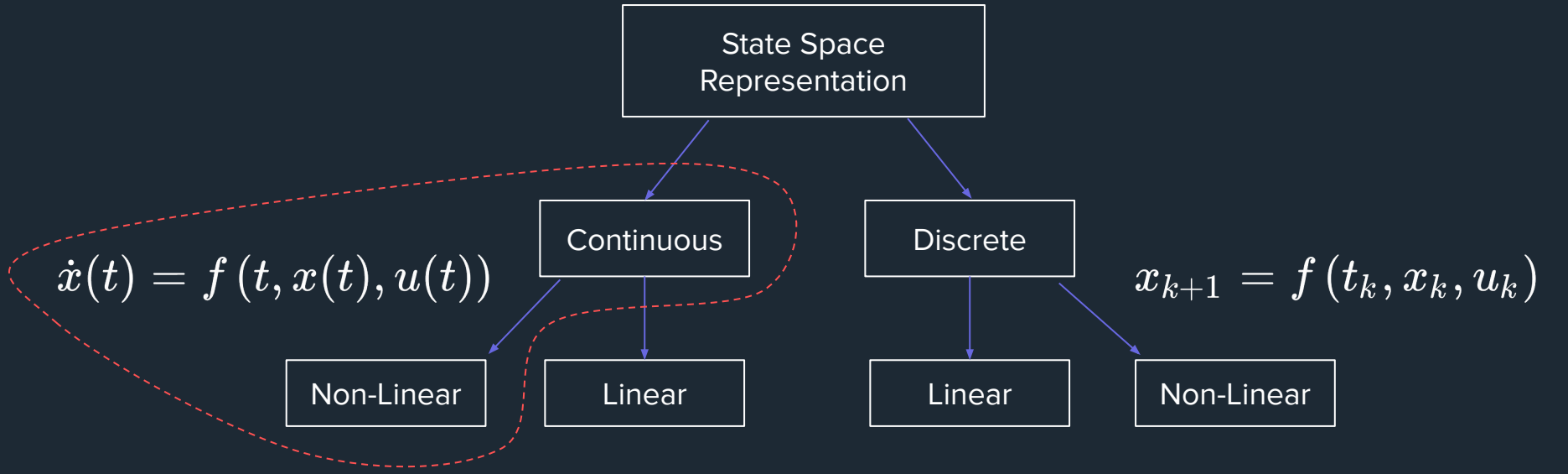
$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$

$$x_{k+1} = f\left(t_k, x_k, u_k\right)$$

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$

$$x_{k+1} = F_k x_k + G_k u_k$$

Mathematical Models

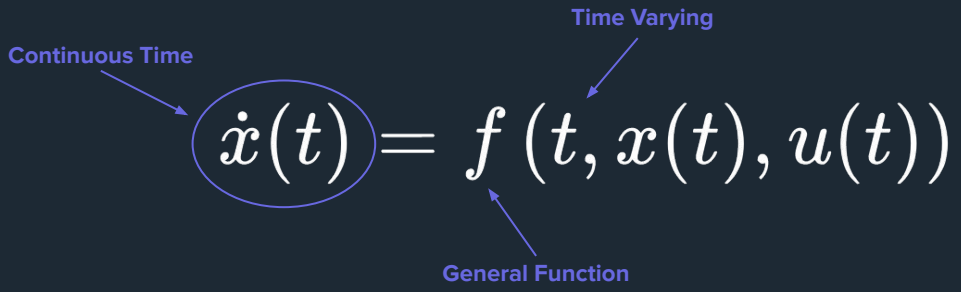


$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$

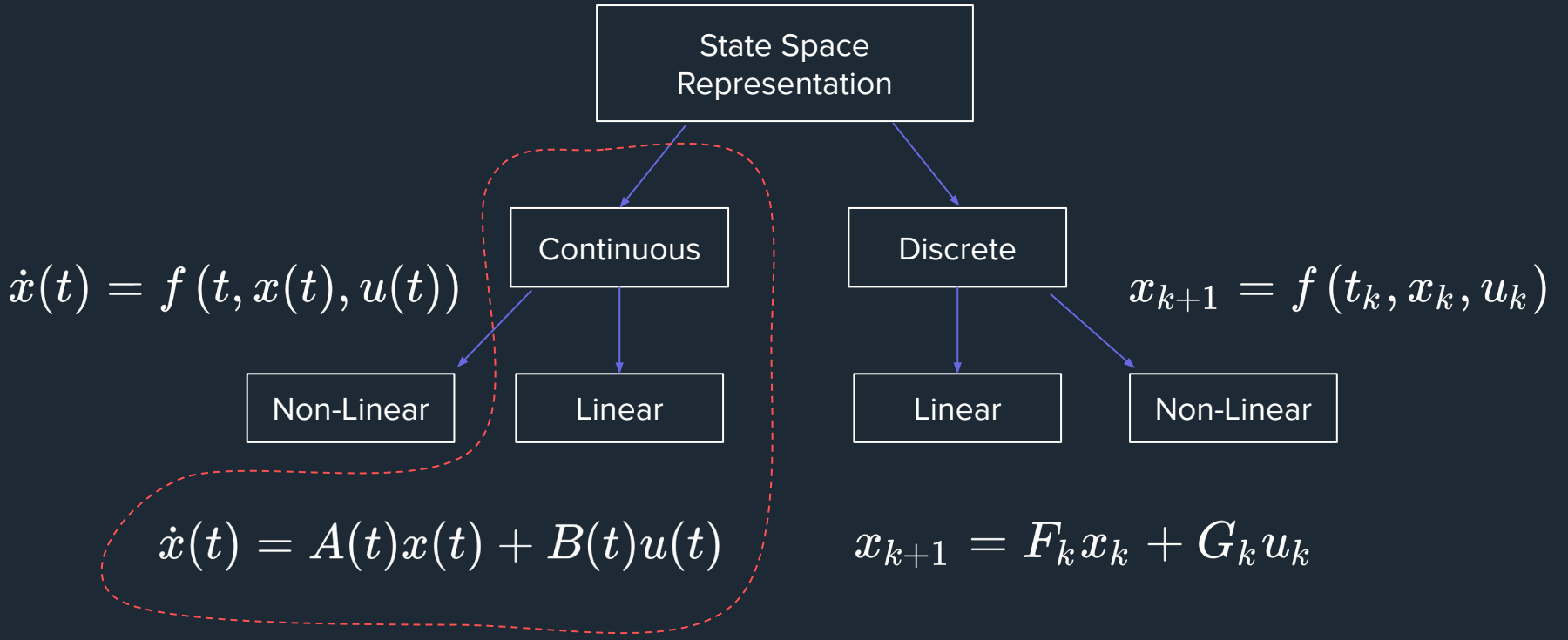
$$x_{k+1} = F_k x_k + G_k u_k$$

Continuous Non-Linear Model

$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$

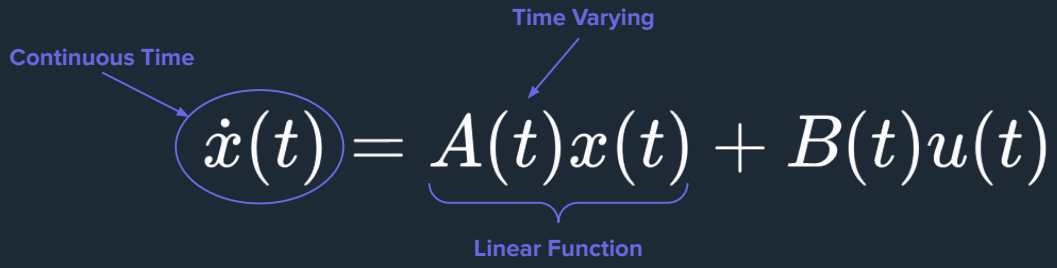


Mathematical Models



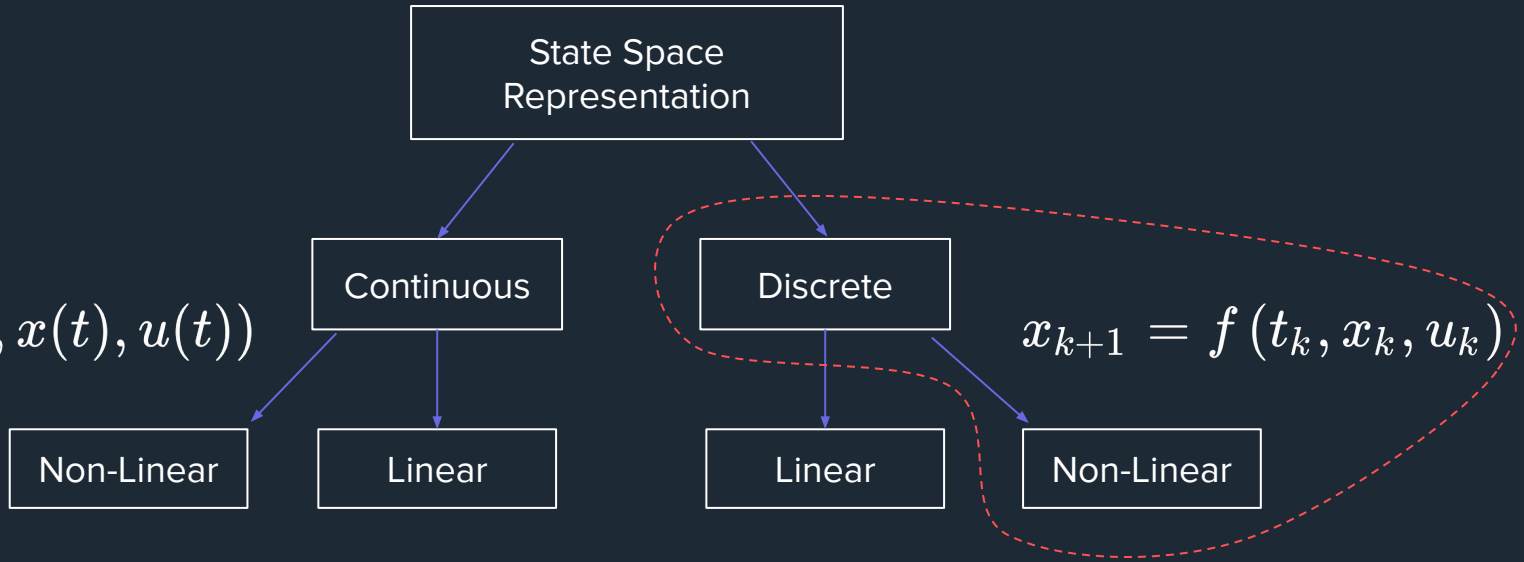
Continuous Linear Model

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$



$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix} = \begin{bmatrix} a_{11}(t) & a_{12}(t) & \dots & a_{1n}(t) \\ a_{21}(t) & a_{22}(t) & \dots & a_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}(t) & a_{n2}(t) & \dots & a_{nn}(t) \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} + \begin{bmatrix} b_{11}(t) & b_{12}(t) & \dots & b_{1r}(t) \\ b_{21}(t) & b_{22}(t) & \dots & b_{2r}(t) \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1}(t) & b_{n2}(t) & \dots & b_{nr}(t) \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_r(t) \end{bmatrix}$$

Mathematical Models



$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$

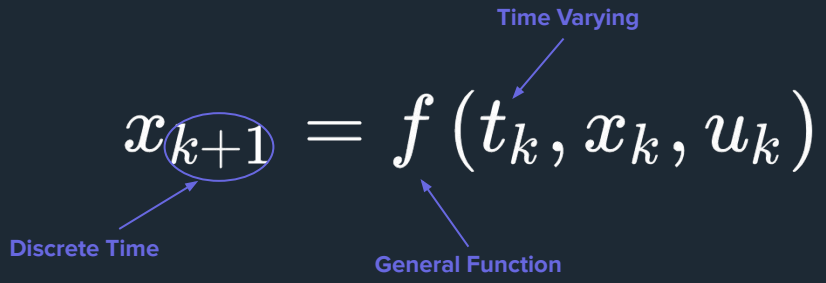
$$x_{k+1} = f\left(t_k, x_k, u_k\right)$$

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$

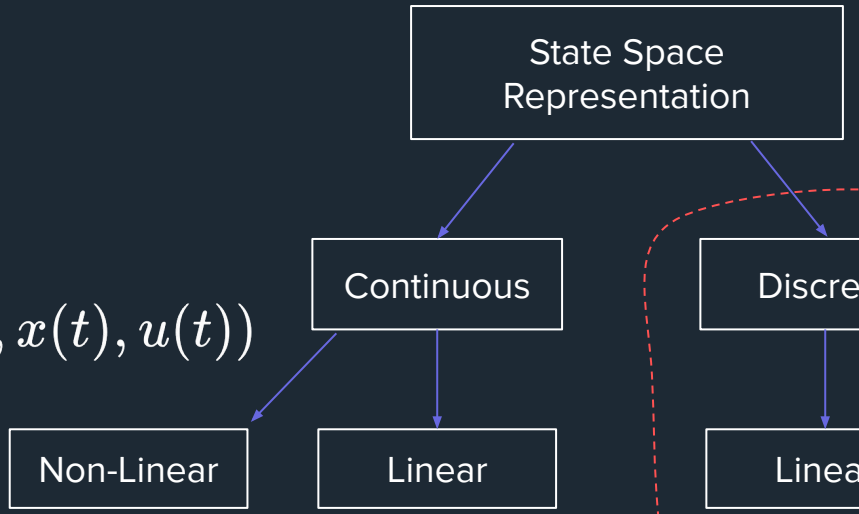
$$x_{k+1} = F_k x_k + G_k u_k$$

Discrete Non-Linear Model

$$x_{k+1} = f\left(t_k, x_k, u_k\right)$$



Mathematical Models



$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$

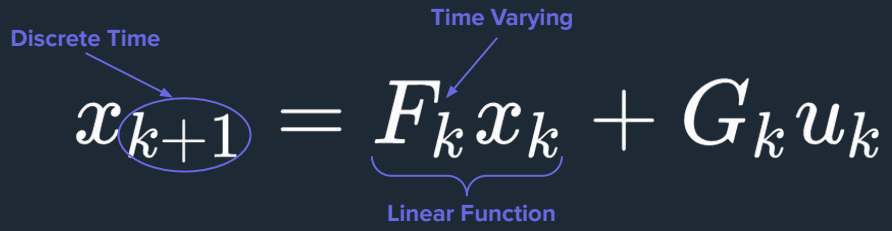
$$x_{k+1} = f\left(t_k, x_k, u_k\right)$$

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$

$$x_{k+1} = F_k x_k + G_k u_k$$

Discrete Linear Model

$$x_{k+1} = F_k x_k + G_k u_k$$



$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{k+1} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{bmatrix}_k \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_k + \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1r} \\ g_{21} & g_{22} & \cdots & g_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nr} \end{bmatrix}_k \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \end{bmatrix}_k$$

Mathematical Models

	Continuous	Discrete
Time Varying		
General	$\dot{x}(t) = f\left(t, x(t), u(t)\right)$	$x_{k+1} = f\left(t_k, x_k, u_k\right)$
Linear	$\dot{x}(t) = A(t)x(t) + B(t)u(t)$	$x_{k+1} = F_k x_k + G_k u_k$
Time Invariant		
General	$\dot{x}(t) = f\left(x(t), u(t)\right)$	$x_{k+1} = f\left(x_k, u_k\right)$
Linear	$\dot{x}(t) = Ax(t) + Bu(t)$	$x_{k+1} = Fx_k + Gu_k$

Steven Dumble, PhD

Technitute_

Linear Dynamic Systems: Discrete Conversions

Steven Dumble, PhD

Continuous to Discrete Conversion

Continuous, Time Invariant, Linear System:

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

$$(A, B) \rightarrow (F, G)$$

Discrete, Time Invariant, Linear System:

$$x_{k+1} = Fx_k + Gu_k$$

Continuous System

Continuous, Time Invariant, Linear System:

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

Solution:

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau$$

Consider a Discrete Time Step:

Current Time Step: $t = t_k$

Previous Time Step: $t_0 = t_{k-1}$

Time Step Size: $\Delta t = t_k - t_{k-1}$

$$x(t_k) = e^{A\Delta t}x(t_{k-1}) + e^{A\Delta t}\int_{0}^{\Delta t} e^{-A\alpha}d\alpha\, Bu(t_{k-1})$$

Continuous to Discrete Conversion

$$x(t_k) = e^{A\Delta t}x(t_{k-1}) + e^{A\Delta t}\int_0^{\Delta t} e^{-A\alpha}d\alpha\, Bu(t_{k-1})$$

$$x_k = Fx_{k-1} + Gu_{k-1}$$

$$\begin{aligned} F &= e^{A\Delta t} \\ G &= F\int_0^{\Delta t} e^{-A\alpha}d\alpha\, B \\ &= F\left[I - e^{-A\Delta t}\right]A^{-1}B \ \text{ if } A^{-1} \text{ exist} \end{aligned}$$

Matrix Exponential

$$e^{At} = \sum_{j=0}^{\infty} \frac{(At)^j}{j!}$$

$$e^{At} = (At)^0 + (At)^1 + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \dots$$

First Order Approximation:

$$e^{At} \approx I + At$$

Steven Dumble, PhD

Technitute_

Linear Dynamic Systems: Simulation of Models

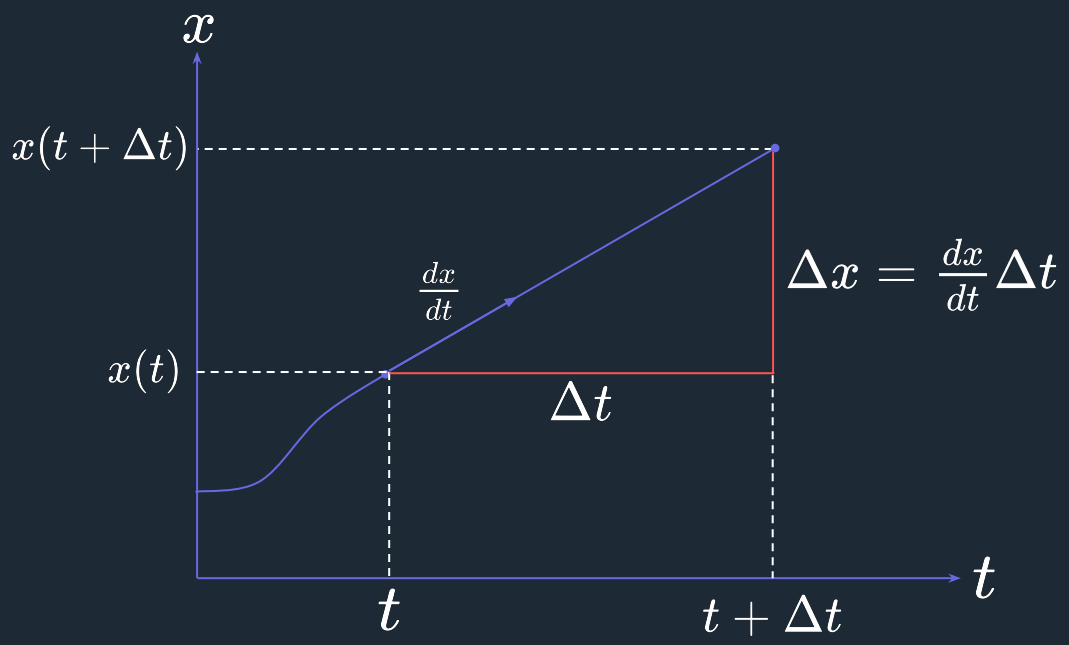
Steven Dumble, PhD

Simulation of Models

- In order to solve the differential equations with respect to time, the integral of the equation needs to be computed.
- This process is called integration, and in simulation it is usually carried out numerically because an exact analytical solution can not usually be found.
- The simulation (or numerical integration) of continuous systems and discrete systems will be explained.

Continuous Time Simulation

$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$



$$x(t + \Delta t) = x(t) + \dot{x}(t)\Delta t$$

Continuous Time Simulation

Euler First Order Integration:

$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$
$$x(t + \Delta t) = x(t) + \dot{x}(t)\Delta t$$
$$\Delta t \to 0$$

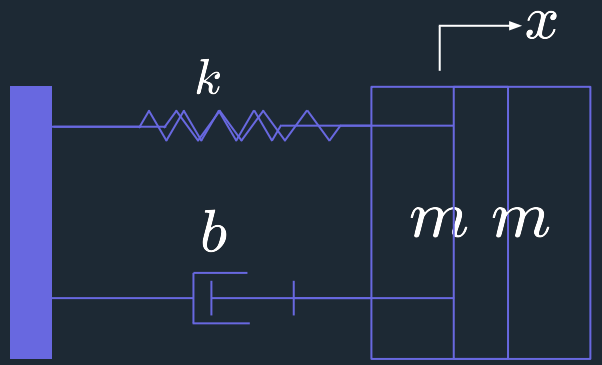
- Simplest Integration Method
- Least Accurate
- Time step must be small to capture dynamics
- Can be numerically unstable (Solution accuracy gets exponentially worse with time)

1) Assume an initial condition
$$x(0)$$
2) Calculate the Equation Rate
$$\dot{x}(t) = f\left(t, x(t), u(t)\right)$$
3) Integrate the Time Step
$$x(t + \Delta t) = x(t) + \dot{x}(t)\Delta t$$
$$t \to t + \Delta t$$
4) Repeat steps 2-3 as needed

2nd Order Mechanical System Example

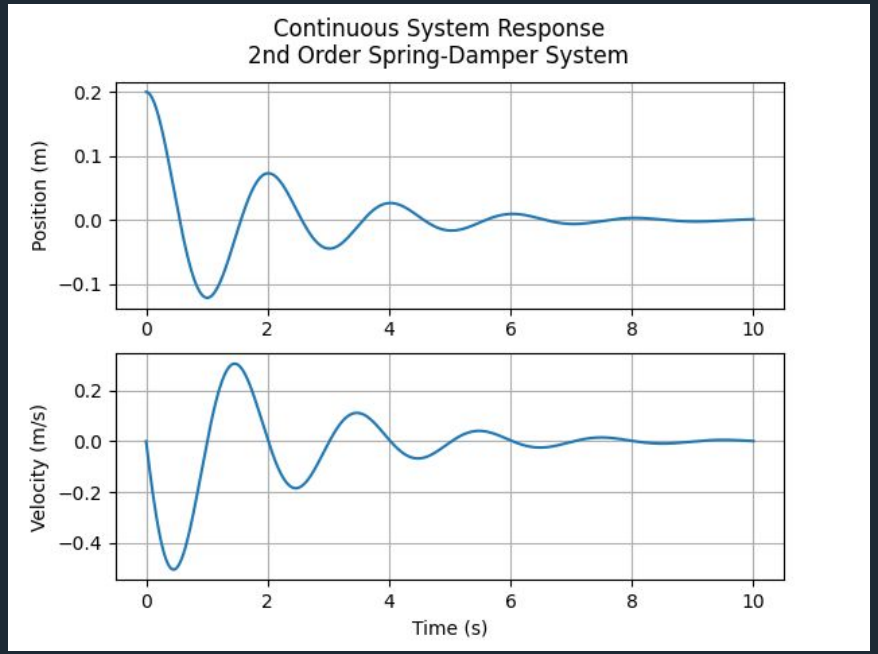
- Consider the Unforced (No input) 2nd Order Mass-Spring-Damper system:

$$\begin{bmatrix} \dot{v} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \frac{-b}{m} & \frac{-k}{m} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ x \end{bmatrix}$$

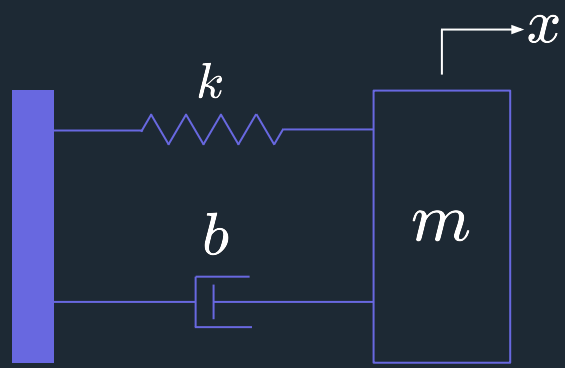


2nd Order Mechanical System Example

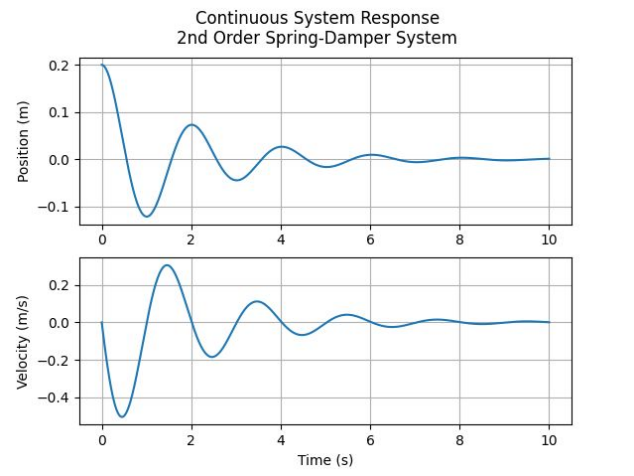
$$\begin{bmatrix} \dot{v} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \frac{-b}{m} & \frac{-k}{m} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ x \end{bmatrix}$$



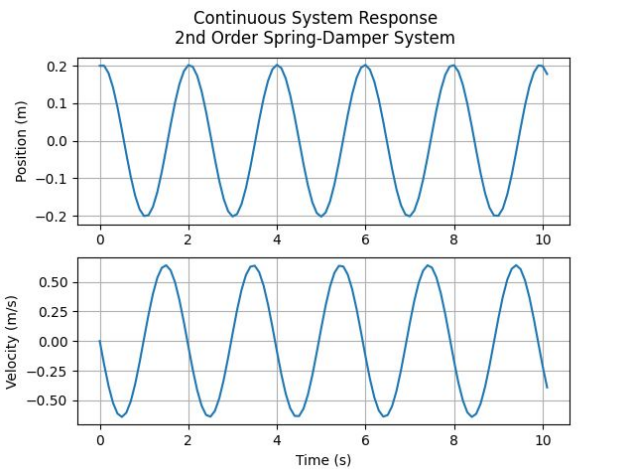
$$m = 1 \text{ kg}$$
$$b = 1.0 \text{ N/(m/s)}$$
$$k = 10.0 \text{ N/m}$$
$$v(0) = 0.0 \text{ m/s}$$
$$x(0) = 0.2 \text{ m}$$



Euler Integration Time Step Size



$\Delta t = 0.0001 sec$



$\Delta t = 0.1 sec$

Discrete Time Simulation

Since the solution is already in discrete time, the simulation of the models is very straight forward, no integration is needed, the recursive solution just needs to be stepped forward from the start of the simulation until the desired time.

1) Assume an initial condition

$$x_0$$

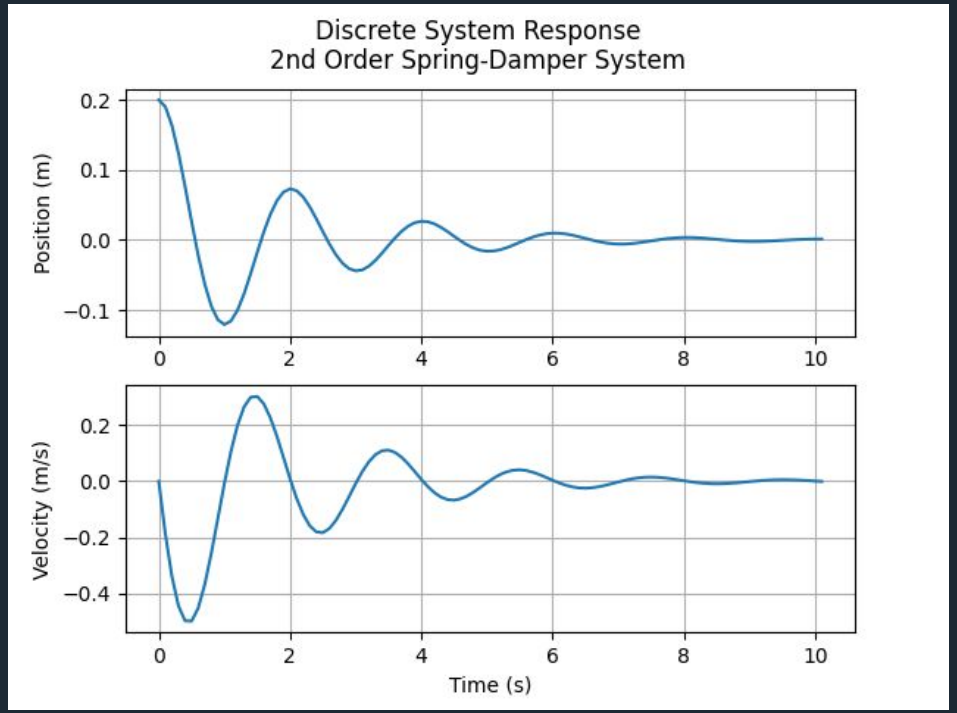
2) Step the Solution

$$x_{k+1} = Fx_k + Gu_k$$

$$k \rightarrow k+1$$

3) Repeat step 2 as needed

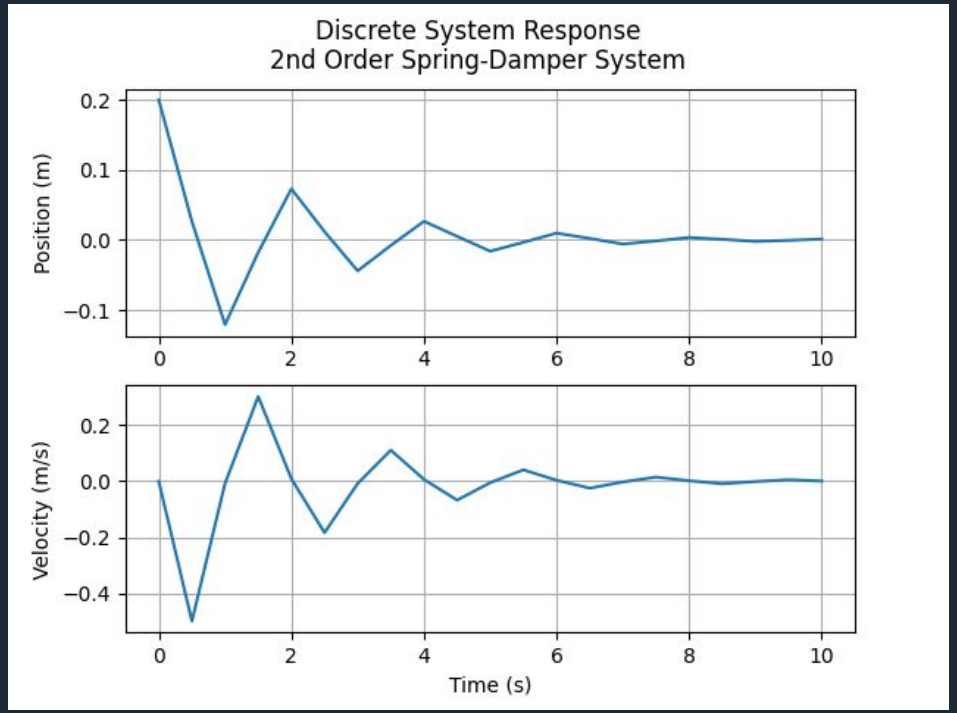
2nd Order Mechanical System Example



$$\Delta t = 0.1$$

$$\begin{bmatrix} v_{k+1} \\ x_{k+1} \end{bmatrix} = \begin{bmatrix} 0.858 & -0.936 \\ 0.094 & 0.952 \end{bmatrix} \begin{bmatrix} v_k \\ x_k \end{bmatrix}$$

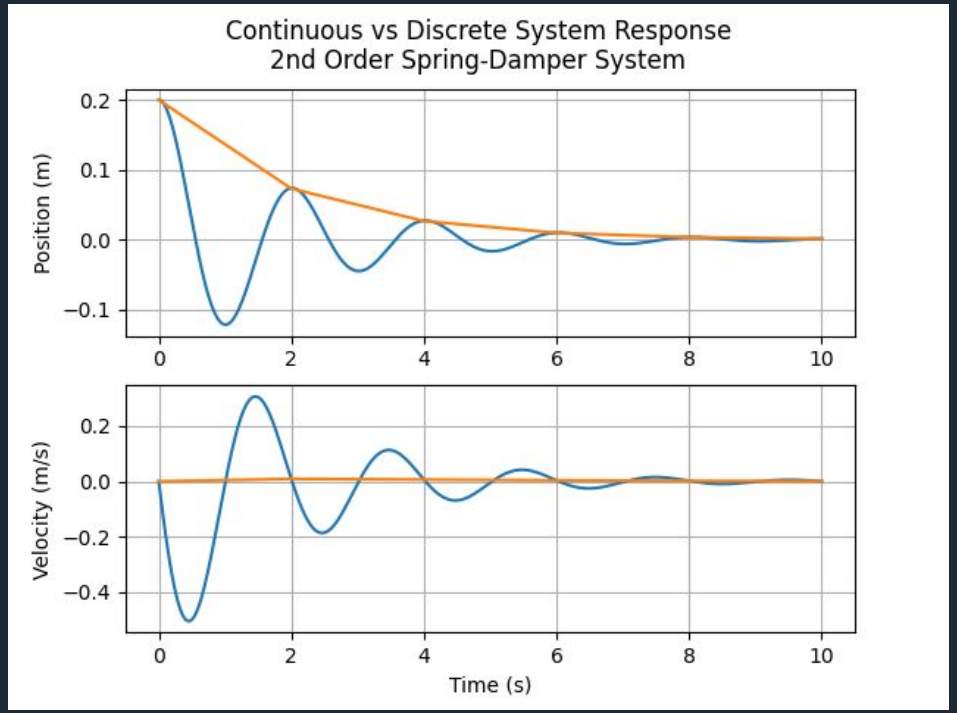
2nd Order Mechanical System Example



$$\Delta t = 0.5$$

$$\begin{bmatrix} v_{k+1} \\ x_{k+1} \end{bmatrix} = \begin{bmatrix} -0.117 & -2.490 \\ 0.249 & 0.132 \end{bmatrix} \begin{bmatrix} v_k \\ x_k \end{bmatrix}$$

2nd Order Mechanical System Example

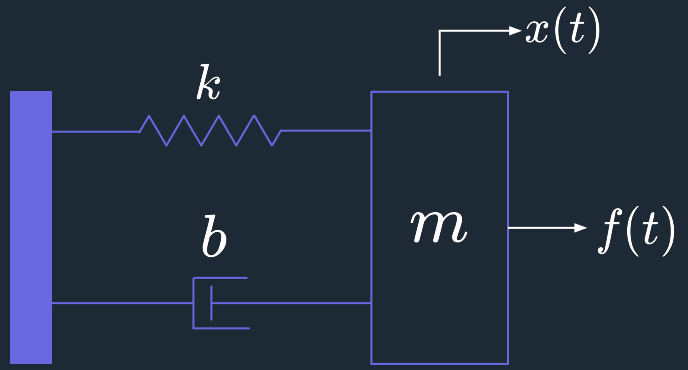


$$\Delta t = 2.0$$

$$\begin{bmatrix} v_{k+1} \\ x_{k+1} \end{bmatrix} = \begin{bmatrix} 0.370 & 0.450 \\ -0.004 & 0.365 \end{bmatrix} \begin{bmatrix} v_k \\ x_k \end{bmatrix}$$

2nd Order Mass-Spring-Damper System

$$\begin{bmatrix} \dot{v} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \frac{-b}{m} & \frac{-k}{m} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ x \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \\ 0 \end{bmatrix} [f]$$



Steven Dumble, PhD

Technitute_

Data Fusion with the Linear Kalman Filter

Introduction Series

Least Squares Estimation

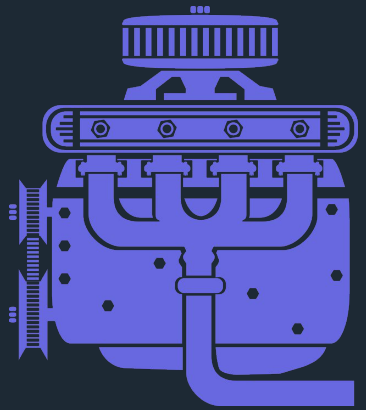
Steven Dumble, PhD

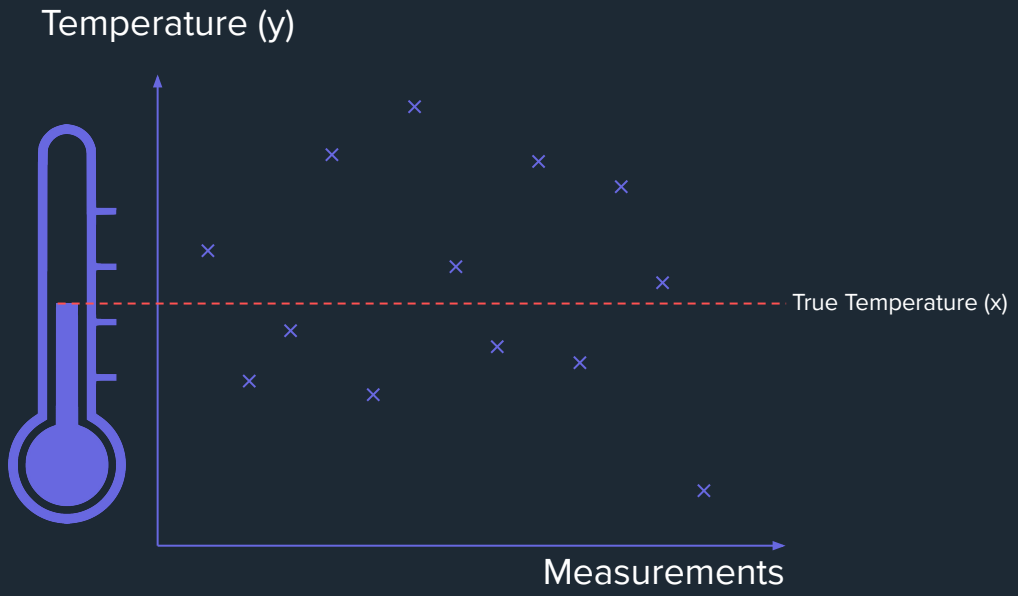
Technitute_

Least Squares Estimation: Estimation of a Constant

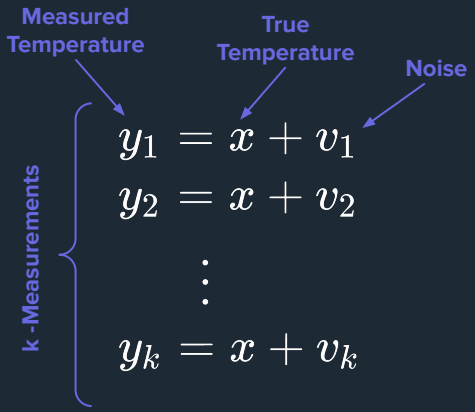
Steven Dumble, PhD

Estimation of a Scalar Constant





Estimation of a Scalar Constant



Noise on the measurement is zero mean, uncorrelated white noise

$$E(V) = 0$$
$$E(v_i v_j^T) = 0$$

$$\bar{y} = \frac{1}{k}\sum_{i=1}^{k} y_i$$

$$\bar{y} = \frac{1}{k}\sum_{i=1}^{k} (x + v_i)$$

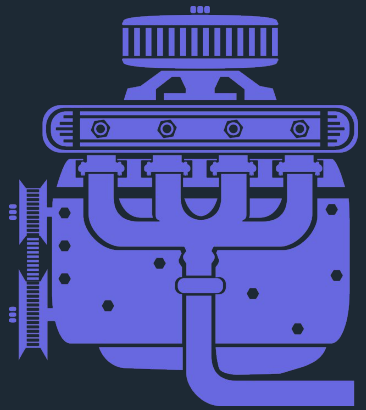
$$\bar{y} = x + \frac{1}{k}\sum_{i=1}^{k} v_i = E(V)$$

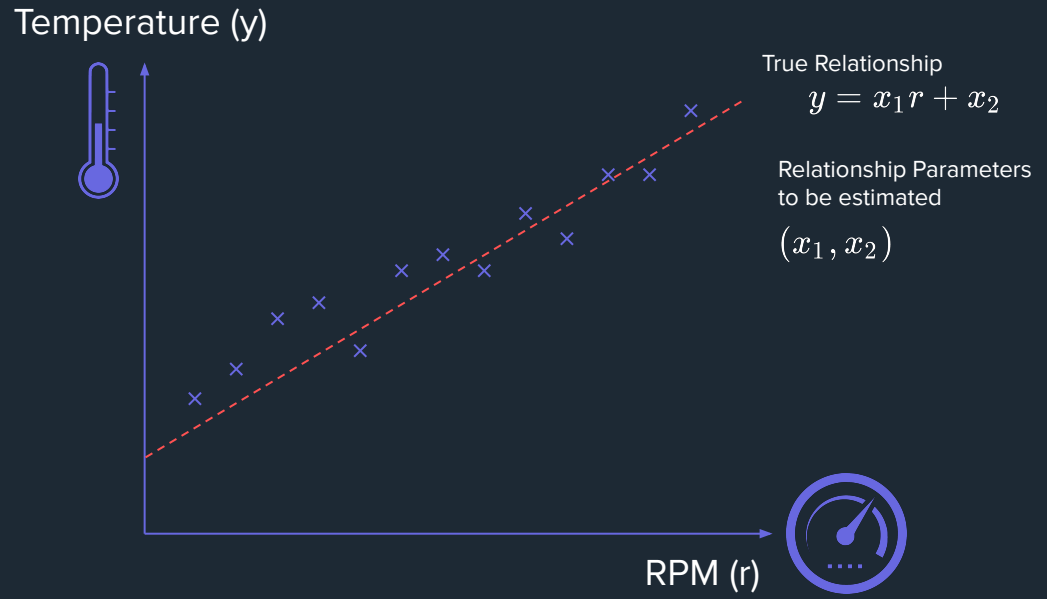
0

The best estimate is given by:

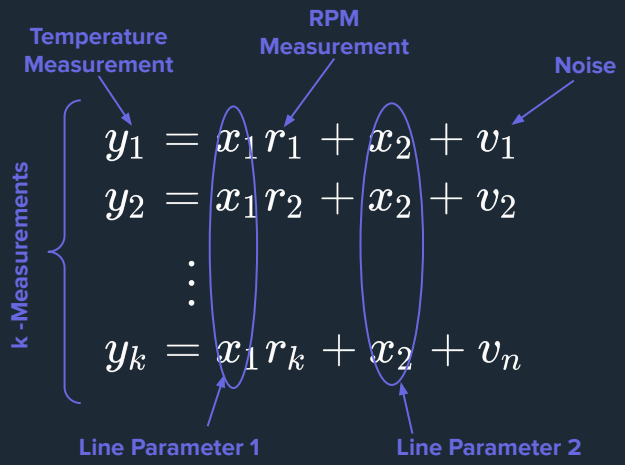
$$\hat{x} = \bar{y}$$

Estimation of a Vector Constant





Estimation of a Vector Constant



$$y = Hx + v$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} r_1 & 1 \\ r_2 & 1 \\ \vdots & \vdots \\ r_k & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}$$

Noise on the measurement is zero mean, uncorrelated white noise

$$E(V) = 0$$
$$E(v_i v_j^T) = 0$$

Estimation of a Vector Constant

$$y = Hx + v$$

$$y - Hx = v$$

Unknown Noise

$$= E(V)$$

Measurement Residual:

$$\epsilon = y - H\hat{x}$$

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_k \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} - \begin{bmatrix} H_{11} & H_{12} & \dots & H_{1n} \\ H_{21} & H_{22} & \dots & H_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{k1} & H_{k2} & \dots & H_{kn} \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_n \end{bmatrix}$$

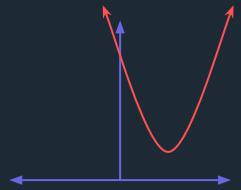
Cost Function:

$$J = \epsilon_1^2 + \epsilon_2^2 + \cdots + \epsilon_k^2$$

$$J = \epsilon^T \epsilon$$

Best Estimate:

$$\hat{x} = \min J$$



Estimation of a Vector Constant

Expand Cost Function:

$$J = \epsilon^T \epsilon$$
$$J = (y - H\hat{x})^T (y - H\hat{x})$$
$$J = y^T y - \hat{x}^T H^T y - y^T H\hat{x} + \hat{x}^T H^T H\hat{x}$$

Find Minimum:

0 = Stationary Point (Potential Minimum)

$$\frac{\partial J}{\partial \hat{x}} = -y^T H - y^T H + 2\hat{x}^T H^T H$$

$$\hat{x} = \left(H^T H\right)^{-1} H^T y$$

Estimation of a Vector Constant

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} r_1 & 1 \\ r_2 & 1 \\ \vdots & \vdots \\ r_k & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}$$

Temperature	RPM (x1000)
65	1
65	2
81	3
92	4
97	5

$$y = \begin{bmatrix} 65 \\ 65 \\ 81 \\ 92 \\ 97 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix}$$

Estimation of a Vector Constant

$$(H^T H) = \begin{bmatrix} 55 & 15 \\ 15 & 5 \end{bmatrix}$$

$$(H^T H)^{-1} = \begin{bmatrix} \frac{1}{10} & -\frac{3}{10} \\ -\frac{3}{10} & \frac{11}{10} \end{bmatrix}$$

$$(H^T H)^{-1} H^T = \begin{bmatrix} -\frac{1}{5} & -\frac{1}{10} & 0 & \frac{1}{10} & \frac{1}{5} \\ \frac{4}{5} & \frac{1}{2} & \frac{1}{5} & -\frac{1}{10} & -\frac{2}{5} \end{bmatrix}$$

$$\hat{x} = (H^T H)^{-1} H^T y = \begin{bmatrix} \frac{91}{10} \\ \frac{527}{10} \end{bmatrix}$$

Temperature vs RPM Relationship:

$$t = 9.1r + 52.7$$

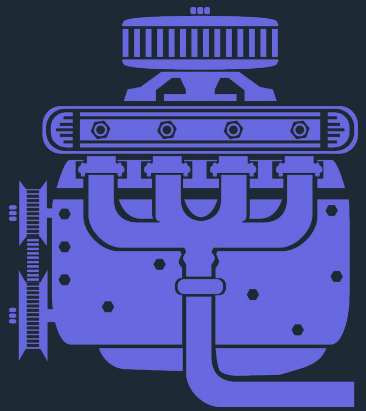
Steven Dumble, PhD

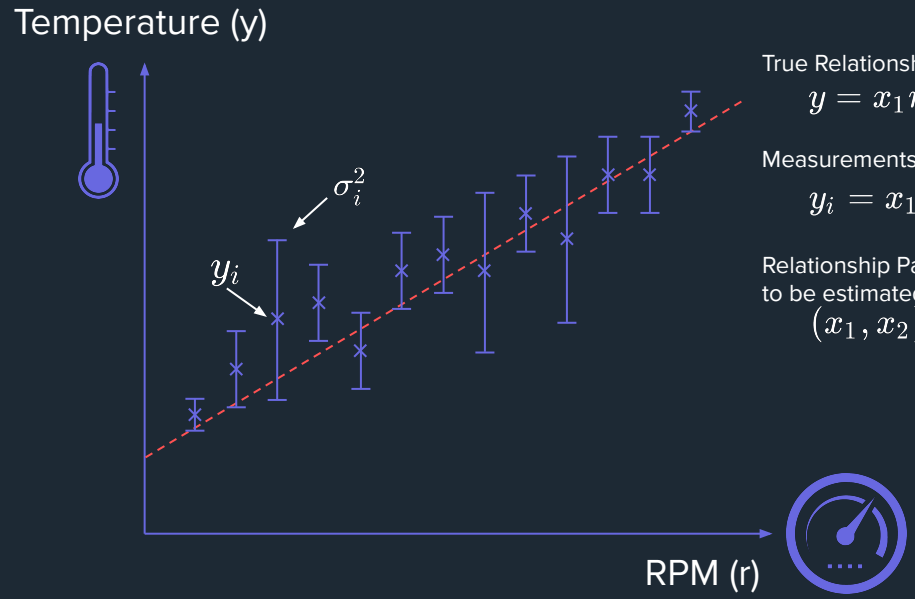
Technitute_

Least Squares Estimation: Weighted Estimation

Steven Dumble, PhD

Weighted Estimation of a Vector Constant





True Relationship

$$y = x_1 r + x_2$$

Measurements

$$y_i = x_1 r + x_2 + v_i$$

Relationship Parameters to be estimated

$$(x_1, x_2)$$

Weighted Estimation of a Vector Constant

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & \dots & H_{1n} \\ H_{21} & H_{22} & \dots & H_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{k1} & H_{k2} & \dots & H_{kn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}$$

$$E(V) = 0$$

$$E(v_i v_j^T) = 0$$

$$E(v_i^2) = \sigma_i^2$$

$$E(vv^T) = R = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_k^2 \end{bmatrix}$$

Noise on the measurement is zero mean, uncorrelated white noise, but each value can have a different variance

Weighted Estimation of a Vector Constant

Problem Model:

$$y = Hx + v$$
$$E(v_i^2) = \sigma_i^2$$
$$E(vv^T) = R$$

Measurement Residual:

$$\epsilon_i = y_i - H_i\hat{x}$$
$$\epsilon = y - H\hat{x}$$

Cost Function:

$$J = \frac{\epsilon_1^2}{\sigma_1^2} + \frac{\epsilon_2^2}{\sigma_2^2} + \cdots + \frac{\epsilon_1^k}{\sigma_k^2} = \epsilon^T R^{-1} \epsilon$$

Best Estimate:

$$\hat{x} = \min J$$

Weighted Estimation of a Vector Constant

Expand Cost Function:

$$J = \epsilon^T R^{-1} \epsilon$$

$$J = (y - H\hat{x})^T R^{-1} (y - H\hat{x})$$

$$J = y^T R^{-1} y - \hat{x}^T H^T R^{-1} y - y^T R^{-1} H\hat{x} + \hat{x}^T H^T R^{-1} H\hat{x}$$

Find Minimum:

0 = Stationary Point (Potential Minimum)

$$\frac{\partial J}{\partial \hat{x}} = -y^T R^{-1} H + \hat{x}^T H^T R^{-1} H$$

$$\hat{x} = \left(H^T R^{-1} H\right)^{-1} H^T R^{-1} y$$

Weighted Estimation of a Vector Constant

Weighted Least Squares Solution:

$$\hat{x} = Ay$$
$$A = \left(H^T R^{-1} H\right)^{-1} H^T R^{-1}$$

Solution Uncertainty:

$$P = ARA^T$$

$$\boxed{P = \left(H^T R^{-1} H\right)^{-1}}$$

Transformation of Uncertainty:

$$f = Ax$$
$$\Sigma_f = A\Sigma_x A^T$$

Estimation of a Vector Constant (Line Fitting)

Temperature y_i	Uncertainty σ_i^2	RPM (x1000) r_i
65	25	1
65	25	2
81	1	3
92	4	4
97	9	5

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} r_1 & 1 \\ r_2 & 1 \\ \vdots & \vdots \\ r_k & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}$$

$$E(vv^T) = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_k^2 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} \quad y = \begin{bmatrix} 65 \\ 65 \\ 81 \\ 92 \\ 97 \end{bmatrix} \quad R = \begin{bmatrix} 25 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 9 \end{bmatrix}$$

$$\hat{x} = \left(H^T R^{-1} H\right)^{-1} H^T R^{-1} y$$

$$\hat{x} = \begin{bmatrix} 9.19 \\ 53.43 \end{bmatrix} \Leftrightarrow y = (9.19)r + 53.43$$

$$P = (H^T R^{-1} H)^{-1}$$

$$P = \begin{bmatrix} 1.24 & -4.01 \\ -4.01 & 13.72 \end{bmatrix}$$

Steven Dumble, PhD

Technitute_

Least Squares Estimation: Recursive Estimation

Steven Dumble, PhD

Least Squares

Versions of Least Squares:

- Least Squares
- Weighted Least Squares
- Recursive Least Squares

$$\hat{x} = \left(H^T H\right)^{-1} H^T y$$

$$\hat{x} = \left(H^T R^{-1} H\right)^{-1} H^T R^{-1} y$$

$$P = \left(H^T R^{-1} H\right)^{-1}$$

Recursive Estimation of a Vector Constant

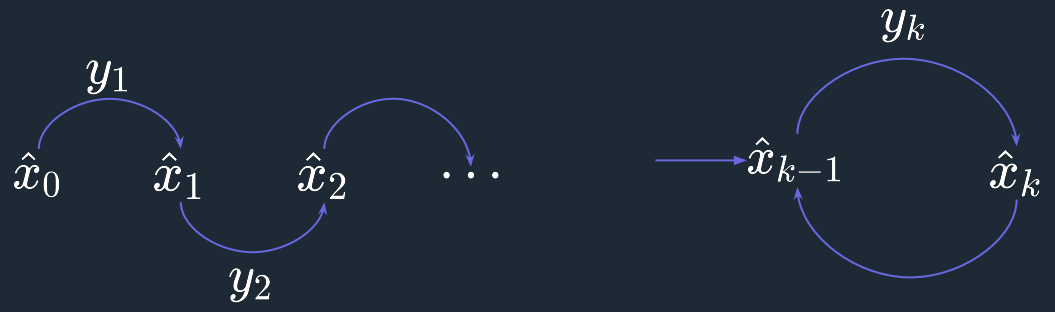
Measurement:

$$y_k = H_k x + v_k$$

$$E(v_k v_k^T) = R_k$$

$$E(V) = 0$$

$$E(v_i v_j^T) = 0$$



Current Best Estimate

Difference between measurement and estimated measurement

$$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k \hat{x}_{k-1})$$

Previous Best Estimate

Correction Gain that makes the minimises the least square cost function

Recursive Estimation of a Vector Constant

Recursive Estimation Error:

$$\epsilon_k = x - \hat{x}_k$$
$$\epsilon_k = x - \hat{x}_{k-1} - K_k(y_k - H_k\hat{x}_{k-1})$$
$$\epsilon_k = \epsilon_{k-1} - K_k(H_k x + v_k - H_k\hat{x}_{k-1})$$
$$\epsilon_k = (I - K_k H_k)\epsilon_{k-1} - K_k v_k$$

Recursive Uncertainty:

$$P_k = \epsilon_k \epsilon_k^T$$
$$P_k = [(I - K_k H_k)\epsilon_{k-1} - K_k v_k][\ldots]^T$$
$$P_k = (I - K_k H_k)P_{k-1}(I - K_k H_k)^T + K_k R_k K_k^T$$

Recursive Estimation of a Vector Constant

Cost Function:

$$J_k = (x_1 - \hat{x}_1)^2 + (x_2 - \hat{x}_2)^2 + \cdots + (x_k - \hat{x}_k)^2$$

$$J_k = \epsilon_1^2 + \epsilon_2^2 + \cdots + \epsilon_k^2$$

$$J_k = \epsilon_k^T \epsilon_k$$

$$J_k = Tr(P_k)$$

Minimizing the solution uncertainty, minimises the cost function

Find Minimum:

0 = Stationary Point (Potential Minimum)

$$\frac{\partial J_k}{\partial K_k} = -2(I - K_k H_k) P_{k-1} H_k^T + 2K_k R_k$$

$$K_k = P_{k-1} H_k^T (H_k P_{k-1} H_k^T + R_k)^{-1}$$

Recursive Estimation of a Vector Constant

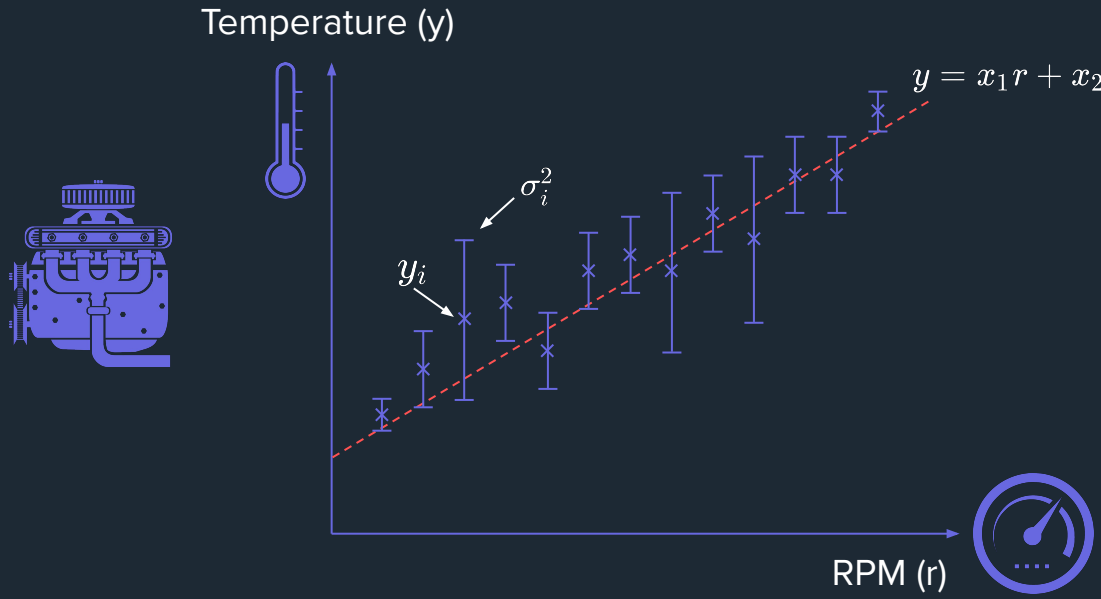
Initial Guess:

$$\hat{x}_0 = E(x)$$
$$P_0 = E[(x - \hat{x}_0)(x - \hat{x}_0)^T]$$

Recursive Least Squares:

$$K_k = P_{k-1} H_k^T (H_k P_{k-1} H_k^T + R_k)^{-1}$$
$$\hat{x}_k = \hat{x}_{k-1} + K_k (y_k - H_k \hat{x}_{k-1})$$
$$P_k = (I - K_k H_k) P_{k-1} (I - K_k H_k)^T + K_k R_k K_k^T$$

Recursive Estimation of a Vector Constant



$$R_i = \sigma_i^2$$
$$H_i = \begin{bmatrix} r_i & 1 \end{bmatrix}$$

$$\hat{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Very large since we initially have no idea

$$P_0 = \begin{bmatrix} 10^4 & 0 \\ 0 & 10^4 \end{bmatrix}$$

Steven Dumble, PhD

Least Squares Estimation Summary

Least Squares:

$$\hat{x} = \left(H^T H\right)^{-1} H^T y$$

Weighted Least Squares:

$$\hat{x} = \left(H^T R^{-1} H\right)^{-1} H^T R^{-1} y$$
$$P = \left(H^T R^{-1} H\right)^{-1}$$

Recursive Least Squares:

$$K_k = P_{k-1} H_k^T (H_k P_{k-1} H_k^T + R_k)^{-1}$$
$$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k \hat{x}_{k-1})$$
$$P_k = (I - K_k H_k) P_{k-1} (I - K_k H_k)^T + K_k R_k K_k^T$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & \dots & H_{1n} \\ H_{21} & H_{22} & \dots & H_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{k1} & H_{k2} & \dots & H_{kn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}$$

$$E(V) = 0$$
$$E(v_i v_j^T) = 0$$
$$E(v_i^2) = \sigma_i^2$$
$$E(vv^T) = R = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_k^2 \end{bmatrix}$$

Technitute_

Data Fusion with the Linear Kalman Filter

Introduction Series

Linear Kalman Filter

Steven Dumble, PhD

Kalman Filter:
What is the Kalman Filter?

Steven Dumble, PhD

What is the Kalman Filter?

- Recursive estimator that solves the ‘linear-quadratic-estimation-problem’.

Estimated State

Noisy Measurements

$$\hat{x} = E(x|z_1, z_2, \cdots, z_k)$$

True State (Unknown)

$$\hat{x} = \min J$$

State Estimation Error

$$\tilde{x} = x - \hat{x}$$

State Estimation Covariance

$$P = [(x - \hat{x})(x - \hat{x})^T]$$

$$J = [(x - \hat{x})^T (x - \hat{x})] = Tr(P)$$

Minimizing the estimation covariance, minimises the cost function

Kalman Filter

Many uses in all branches of engineering and even in other fields such as finance.

Main application are:

- Guidance, Navigation and Control of Vehicle (Cars, Ships, Aircraft, Spacecraft).
- Robotics and Manufacturing
- Any time-domain series analysis (such as signal processing, economics, stock market prediction, finance, etc...)

Rudolf Emil Kalman

- Born in Budapest in 1930.
- Immigrated to the USA during WW2.
- Bachelor and Masters in Electrical Engineering at MIT.
- Lecturer at Columbia University.
- First practical used of the Kalman Filter was by Ames Research Center at NASA -> Used in the Apollo spacecraft trajectory estimation and control system.

Steven Dumble, PhD

Kalman Filter:
Types of the Kalman Filter

Steven Dumble, PhD

Kalman Filter Types

- Linear Kalman Filter (KF)
- Extended Kalman Filter (EKF)
- Unscented Kalman Filter (UKF)

Linear Kalman Filter

- Assumes Linear Discrete or Continuous System Dynamics
- Uses Linear Covariance Prediction/Update Equations
- Does not work with non-linear systems

Extended Kalman Filter

- Assumes Non-Linear Discrete or Continuous System Dynamics
- Uses a Linear Covariance Prediction/Update Equation, but the linear relationship is approximated from the non-linear dynamics
- Works well with non-linear systems that are smooth and the estimated state is close to the true state

Unscented Kalman Filter

- Assumes Non-Linear Discrete or Continuous System Dynamics
- Uses the non-linear system to calculate the Prediction/Update Equation, making it a better approximation than the EKF. (Rather than linearizing the system dynamics, it calculates a linear approximation of the uncertainty coverances).
- Works on non-linear systems that are slightly more non-linear than what the EKF can handle, since it is better at approximating the non-linear systems.

Steven Dumble, PhD

Technitute_

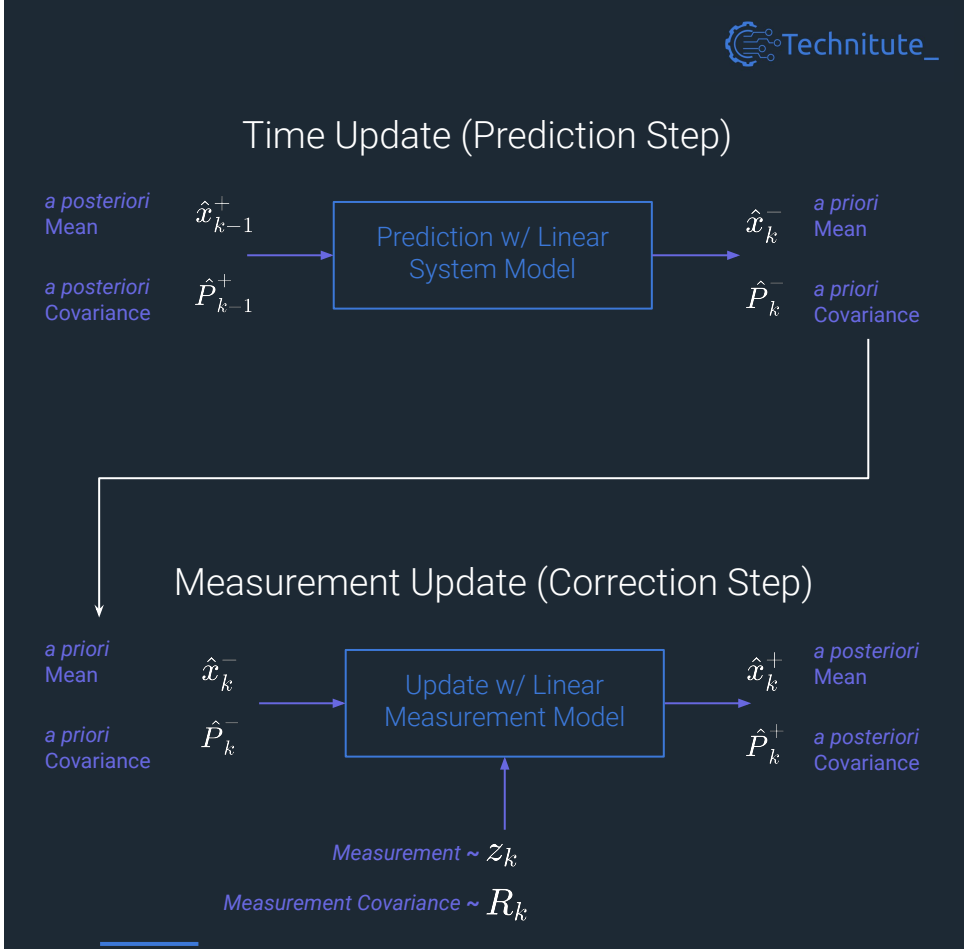
Kalman Filter:
How the Kalman Filter works

Steven Dumble, PhD

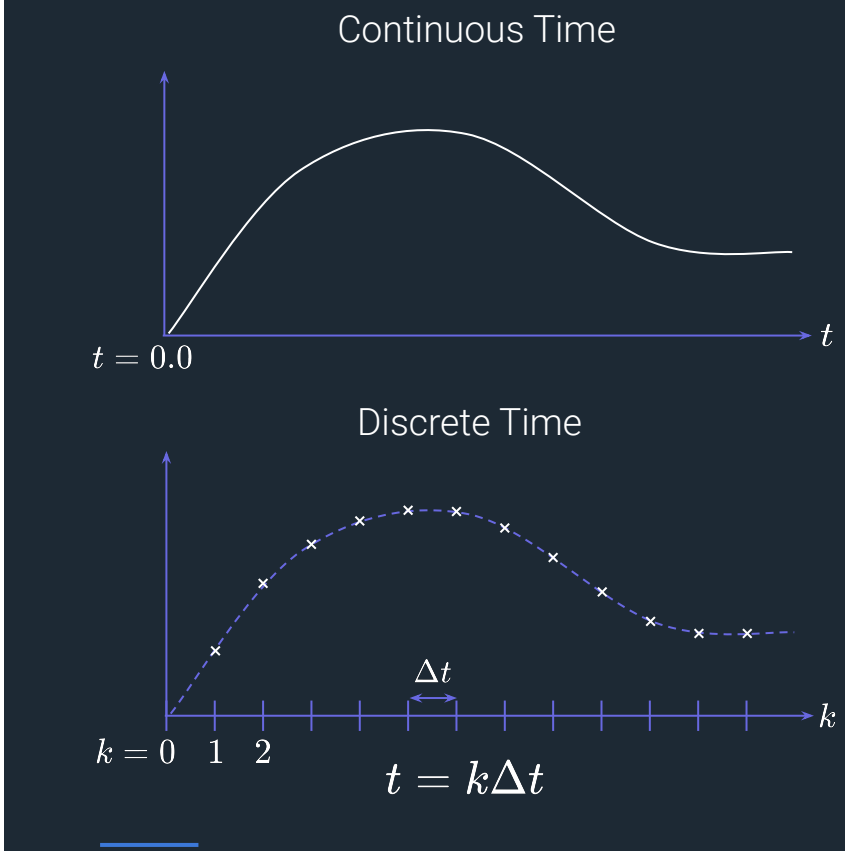
Full Portrait Intro

Steven Dumble, PhD

Video


Technitute_
Time Update (Prediction Step)
a posteriori Mean
$\hat{x}_{k-1}^{+}$
a posteriori Covariance
$\hat{P}_{k-1}^{+}$
Prediction w/ Linear System Model
$\hat{x}_{k}^{-}$
a priori Mean
$\hat{P}_{k}^{-}$
a priori Covariance
Measurement Update (Correction Step)
a priori Mean
$\hat{x}_{k}^{-}$
a priori Covariance
$\hat{P}_{k}^{-}$
Update w/ Linear Measurement Model
$\hat{x}_{k}^{+}$
a posteriori Mean
$\hat{P}_{k}^{+}$
a posteriori Covariance
Measurement ~ z_k
Measurement Covariance ~ R_k

Video



Steven Dumble, PhD

Discrete Time System Model

Process Model: $x_k = \mathbf{F}_{k-1} x_{k-1} + \mathbf{G}_{k-1} u_{k-1} + \mathbf{L}_{k-1} w_{k-1}$

Measurement Model: $z_k = \mathbf{H}_k x_k + \mathbf{M}_k v_k$

- $x_k \sim$ State Vector
- $u_k \sim$ Control Input Vector
- $w_k \sim$ Process Model Noise Vector
- $v_k \sim$ Measurement Noise Vector

- $\mathbf{F}_k \sim$ State Transition Matrix
- $\mathbf{G}_k \sim$ Control Input Matrix
- $\mathbf{H}_k \sim$ Measurement Matrix
- $\mathbf{L}_k \sim$ Process Model Noise Sensitivity Matrix
- $\mathbf{M}_k \sim$ Measurement Model Noise Sensitivity Matrix

Discrete Time System Model

Process Model: $x_k = \mathbf{F}_{k-1} x_{k-1} + \mathbf{G}_{k-1} u_{k-1} + \mathbf{L}_{k-1} w_{k-1}$

Measurement Model: $z_k = \mathbf{H}_k x_k + \mathbf{M}_k v_k$

$w_k \sim N(0, \mathbf{Q}_k)$

$v_k \sim N(0, \mathbf{R}_k)$

Gaussian Distribution with **zero mean** and given **covariance matrix**

$E(w_k w_j^T) = \mathbf{Q}_k \delta_{k-j}$

$E(v_k v_j^T) = \mathbf{R}_k \delta_{k-j}$

Not correlated with time

$E(w_k v_j^T) = 0$

Process noise and Measurement noise are independant

Video

- Our goal is to estimate the state of the system based on our knowledge of the dynamics and the availability of noisy measurements.

a posteriori Estimated State

Noisy Measurements

$$\hat{x}_k^+ = E(x_k | z_1, z_2, \cdots, z_k)$$

True State (Unknown)

a priori Estimated State

$$\hat{x}_k^- = E(x_k | z_1, z_2, \cdots, z_{k-1})$$

Initial State: $\hat{x}_0^+ = E(x_0)$

Video

- The Kalman Filter also uses the covariance of the estimation error to probabilistically find the updated state estimate.

a priori
Estimated Covariance

$$\mathbf{P}_k^- = E\left[\left(x_k - \hat{x}_k^-\right)\left(x_k - \hat{x}_k^-\right)^T\right]$$

a posteriori
Estimated Covariance

$$\mathbf{P}_k^+ = E\left[\left(x_k - \hat{x}_k^+\right)\left(x_k - \hat{x}_k^+\right)^T\right]$$

Estimation Error: $\tilde{x}_k = x_k - \hat{x}_k$

Linear Kalman Filter

- We will develop the equations of how the Kalman Filter calculates the estimated state and covariance in the next series of videos

2D Tracker Example: Simulation Framework

Steven Dumble, PhD

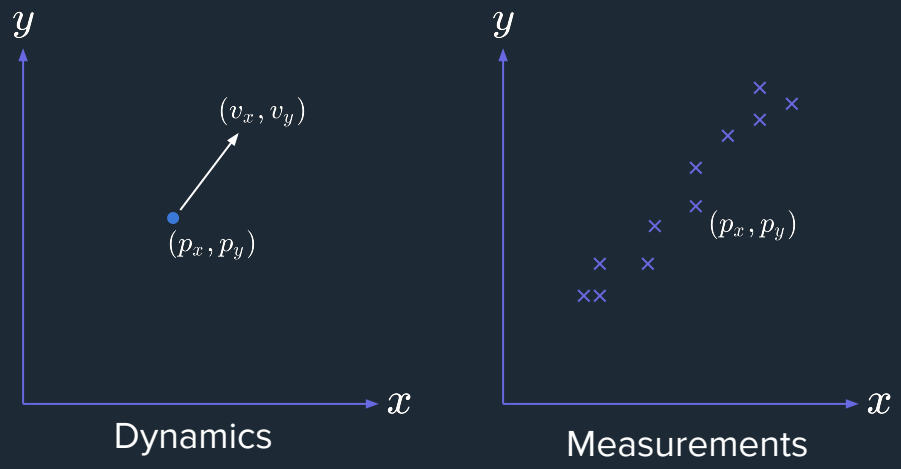
2D Tracking Filter

- Goal: To estimate the position and velocity of a moving object based on **remote** measurements.

- Many Applications:
  - Radar Aircraft Tracking
  - Obstacle Tracking Self-driving Cars
  - Object Tracking in Images
  - Surveillance / Military
  - Robotics
  - GPS Aided Navigation and Guidance

2D Tracking Filter

- Problem:
  - Assume an object travels at a constant speed in the 2D X-Y plane [Process Model]
  - Assume we get position measurements of the objects location (i.e. Radar) [Sensor Model]



2D Tracking Filter Python Simulation

```
Linear Kalman Filter
├── kfsims
│   ├── kfmodels.py
│   ├── kftracker2d.py
│   ├── tracker2d.py
│   ├── vehiclemodel2d.py
│   └── ...
├── assignment1_prediction.py
├── assignment1_prediction_answer.py
├── assignment1_update.py
├── assignment1_update_answer.py
├── assignment1_initial_conditions.py
├── assignment1_initial_conditions_answer.py
├── assignment1_answer.py
└── ...
```

2D Tracking Filter Python Simulation

Linear Kalman Filter
- kfsims
  - kfmodels.py
  - kftracker2d.py
  - tracker2d.py
  - vehiclemodel2d.py
  - ...
- assignment1_prediction.py
- assignment1_prediction_answer.py
- assignment1_update.py
- assignment1_update_answer.py
- assignment1_initial_conditions.py
- assignment1_initial_conditions_answer.py
- assignment1_answer.py
- ...

```python
# Kalman Filter Model
class KalmanFilterModel(KalmanFilterBase):

    def initialise(self, time_step):

        # Define a np.array 4x1 with the initial state (px,py,vx,vy)
        #self.state =

        # Define a np.array 4x4 for the initial covariance
        #self.covariance =

        # Setup the Model F Matrix
        #self.F =

        # Set the Q Matrix
        #self.Q =

        return
```

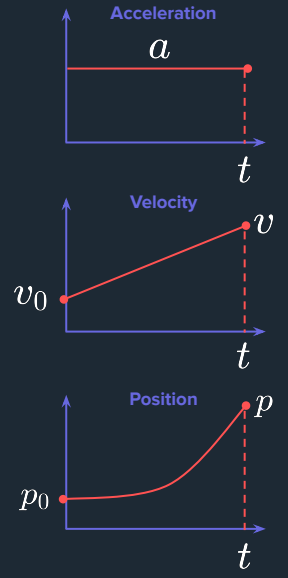
2D Tracker Example: Process Model

2D Tracking System Model

- Derive the System Process model from the Equations of Motion:

$$a = \frac{dv}{dt} = \frac{d^2p}{dt^2}$$

Acceleration
Velocity
Position



- Assuming **constant** acceleration, the relations are found:

$$v = v_0 + at$$

$$p = p_0 + tv_0 + \frac{1}{2}at^2$$

2D Tracking System Model

$$v = v_0 + at$$
$$p = p_0 + tv_0 + \frac{1}{2}at^2$$

- Write the Equations of Motion in Matrix Form:

$$\begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ v_0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}t^2 \\ t \end{bmatrix} a$$

- Convert to Discrete Time: $t = \Delta t$

$$\begin{bmatrix} p \\ v \end{bmatrix}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} [a]$$

2D Tracking System Model

State Vector: $x_k = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$

Input Vector: ~~$u_k = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$~~

Discrete Process Model:

$$x_k = \mathbf{F}x_{k-1} + \cancel{\mathbf{G}u_{k-1}}$$

$$\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_k = \underbrace{\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_{k-1} + \cancel{\underbrace{\begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}}_{\mathbf{G}}} \begin{bmatrix} a_x \\ a_y \end{bmatrix}_{k-1}$$

2D Tracking System Model

State Vector: $x_k = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$

Noise Vector: $w_k = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$ $\quad a_x \sim N(0, \sigma_{a_x}^2)$ $\quad a_y \sim N(0, \sigma_{a_y}^2)$

Discrete Process Model:

$$x_k = \mathbf{F}x_{k-1} + \mathbf{L}w_{k-1}$$

$$\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_k = \underbrace{\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_{k-1} + \underbrace{\begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}}_{\mathbf{L}} \begin{bmatrix} a_x \\ a_y \end{bmatrix}_{k-1}$$

2D Tracking System Model

Discrete Process Model:

$$x_k = \mathbf{F}x_{k-1} + \mathbf{L}w_{k-1}$$

$$\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_k = \underbrace{\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_{k-1} + \underbrace{\begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}}_{\mathbf{L}} \begin{bmatrix} a_x \\ a_y \end{bmatrix}_{k-1}$$

$$a_x \sim N(0, \sigma_{a_x}^2)$$
$$a_y \sim N(0, \sigma_{a_y}^2)$$

Kalman Filter: Prediction Step

Steven Dumble, PhD

Prediction Step (State)

- We being the estimation step with the initial condition $\hat{x}_0^+$ which is the best estimate of the initial state of the system.
- We want to propagate time forward to find $\hat{x}_1^- = E(x_1)$, so can use the state space model of the system:

$$\hat{x}_1^- = \mathbf{F}_0 \hat{x}_0^+ + \mathbf{G}_0 u_0$$

or in general:

$$\hat{x}_k^- = \mathbf{F}_{k-1} \hat{x}_{k-1}^+ + \mathbf{G}_{k-1} u_{k-1}$$

Prediction Step (State)

- This is the prediction step or time update step. We use the process model of the system we want to estimate to predict the estimated state (mean of the gaussian of the probability density function) forward in time.

State Prediction Step:

$$\hat{x}_k^- = \mathbf{F}_{k-1}\hat{x}_{k-1}^+ + \mathbf{G}_{k-1}u_{k-1}$$

Prediction Step (Covariance)

- But we also want to propagate the covariance forward in time as well. To begin with $\mathbf{P}_0^+$ is the initial covariance for the state estimate of x_0
  - Exact knowledge of x_0 then $\mathbf{P}_0^+ = 0$ (there is no uncertainty)
  - No knowledge of x_0 then $\mathbf{P}_0^+ \to \infty$ (there is large uncertainty)
- But we usually know an approximate value (or good assumption) for the value of x_0 so we set $\mathbf{P}_0^+$ to be:

$$\mathbf{P}_0^+ = E\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T\right]$$

Prediction Step (Covariance)

- So given $\mathbf{P}_0^+$, we can propagate this uncertainty with time using the linear transformation:

$$\mathbf{P}_1^- = \mathbf{F}_0 \mathbf{P}_0^+ \mathbf{F}_0^T$$

- But this does not change the amount of the uncertainty in the system, it just shifts it between the states as the system dynamics dictate.

Process Model: $x_k = \mathbf{F}_{k-1} x_{k-1} + \mathbf{G}_{k-1} u_{k-1} + \mathbf{L}_{k-1} w_{k-1}$

- We need to account for the **new** uncertainty in the state due to the process model noise!

Prediction Step (Covariance)

- To account for new uncertainty in the system due to the process model noise we add a new covariance term:

$$\mathbf{P}_1^- = \mathbf{F}_0 \mathbf{P}_0^+ \mathbf{F}_0^T + \mathbf{L}_0 \mathbf{Q}_0 \mathbf{L}_0^T$$

- It is usually assumed that the process noise is additive, so that $\mathbf{L}_k = \mathbf{I}$, which means that the equation can be simplified to:

$$\mathbf{P}_1^- = \mathbf{F}_0 \mathbf{P}_0^+ \mathbf{F}_0^T + \mathbf{Q}_0$$

Prediction Step

Predicted (*a priori*) state estimate:

$$\hat{x}_k^- = \mathbf{F}_{k-1}\hat{x}_{k-1}^+ + \mathbf{G}_{k-1}u_{k-1}$$

Predicted (*a priori*) covariance estimate:

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

2D Tracker Example: Prediction Step

Steven Dumble, PhD

2D Tracking System Model

Process Model: $x_k = \mathbf{F}x_{k-1} + \mathbf{L}w_{k-1}$

Process Noise Model: $\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}^+\mathbf{F}^T + \mathbf{L}\mathbf{Q}\mathbf{L}^T$

$$w_k \sim N(0, \mathbf{Q})$$

$$\mathbf{Q} = \begin{bmatrix} \sigma_{a_x}^2 & 0 \\ 0 & \sigma_{a_y}^2 \end{bmatrix}$$

$$\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_k = \underbrace{\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_{k-1} + \underbrace{\begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}}_{\mathbf{L}} \begin{bmatrix} a_x \\ a_y \end{bmatrix}_{k-1}$$

2D Tracking System Model

Process Model: $x_k = \mathbf{F}x_{k-1} + \mathbf{L}w_{k-1}$

Process Noise Model: $\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}^+\mathbf{F}^T + \mathbf{L}\mathbf{Q}\mathbf{L}^T$

$$w_k \sim N(0, \mathbf{Q})$$

$$\mathbf{Q} = \begin{bmatrix} \sigma_{a_x}^2 & 0 \\ 0 & \sigma_{a_y}^2 \end{bmatrix}$$

Not Valid Distribution

$$\mathbf{L}\mathbf{Q}\mathbf{L}^T = \begin{bmatrix} \frac{1}{4}\Delta t^4\sigma_{a_x}^2 & 0 & \frac{1}{2}\Delta t^3\sigma_{a_x}^2 & 0 \\ 0 & \frac{1}{4}\Delta t^4\sigma_{a_y}^2 & 0 & \frac{1}{2}\Delta t^3\sigma_{a_y}^2 \\ \frac{1}{2}\Delta t^3\sigma_{a_x}^2 & 0 & \Delta t^2\sigma_{a_x}^2 & 0 \\ 0 & \frac{1}{2}\Delta t^3\sigma_{a_y}^2 & 0 & \Delta t^2\sigma_{a_y}^2 \end{bmatrix}$$

2D Tracking System Model

Process Model: $\boldsymbol{x_k = \mathbf{F} x_{k-1} + \mathbf{L} w_{k-1}}$

Process Noise Model: $\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}^+\mathbf{F}^T + \hat{\mathbf{Q}}\mathbf{Q}\mathbf{L}^T$

$$\sigma_a = \sigma_{a_x} = \sigma_{a_y}$$

$$w_k \sim \mathbf{L} \cdot N(0, \sigma_a^2)$$

$$\hat{\mathbf{Q}} = \sigma_a^2 \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix}$$

Assignment 1: Prediction

Implement the **Kalman Filter Prediction Equations** and the **2D Tracking Process Model.**

1. **Step 1)** Open the python file '**assignment1_prediction.py**'
    a. Run the simulation as is. See that the object starts at the origin (px,py) = (0,0) and moves at a 45 deg angle at 10 m/s (vx,vy) = (7.07, 7.07)

Assignment 1: Prediction

Step 2) Setup the initial state and covariance

a. Assume initial position is (0,0) and initial velocity is (7.07, 7.07)

b. Assume no initial uncertainty (Zero matrix)

```
def initialise(self, time_step):

    # Define a np.array 4x1 with the initial state (px,py,vx,vy)
    #self.state =

    # Define a np.array 4x4 for the initial covariance
    #self.covariance =
```

$$x = [0, 0, 7.07, 7.07]^T$$

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Assignment 1: Prediction

Step 3) Setup the model F and Q Matrices

a. Use the time step and define the F process model matrix
b. Define the Q matrix as a function of a variable accel_std
c. Assume the process model noise acceleration stdev is zero initially

```
# Setup the Model F Matrix
#self.F =

# Set the Q Matrix
#self.Q =
```

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

$$\mathbf{Q} = \sigma_a^2 \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix}$$

Assignment 1: Prediction

Step 4) Implement the Kalman Filter Prediction Step Equations

a. State Prediction

b. Covariance Propagation

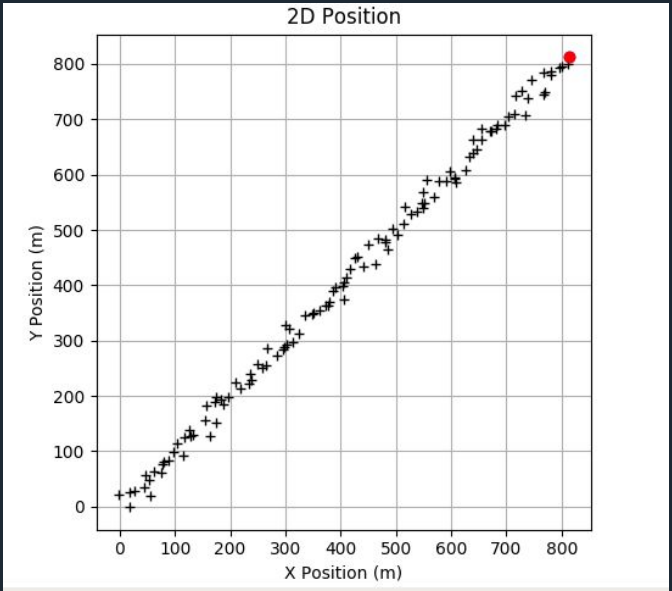
```python
def prediction_step(self):
    # Make Sure Filter is Initialised
    if self.state is not None:
        x = self.state
        P = self.covariance

        # Calculate Kalman Filter Prediction

        # State Prediction: x_predict = F * x
        #x_predict =

        # Covariance Prediction: P_predict = F * P * F' + Q
        #P_predict =

        # Save Predicted State
        self.state = x_predict
        self.covariance = P_predict

    return
```

$$x_k = \mathbf{F}x_{k-1}$$
$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}^+\mathbf{F}^T + \mathbf{Q}$$

Assignment 1: Prediction

Step 5) Run the Simulation

a. Check that the Prediction follows the truth fairly closely



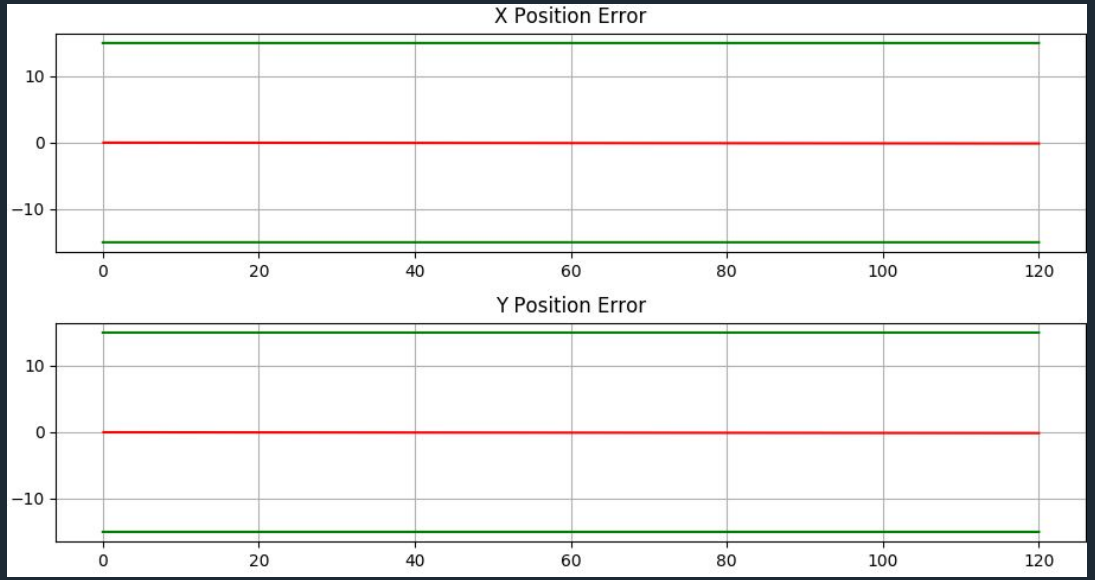

```
Position Mean Squared Error: 0.010959819397519322 (m)^2
Velocity Mean Squared Error: 2.2804443600974467e-06 (m/s)^2
```

Assignment 1: Prediction

Step 6) Check the Position Covariance Prediction is Working Correctly

a. Set the initial position x and y covariance to be (5)^2

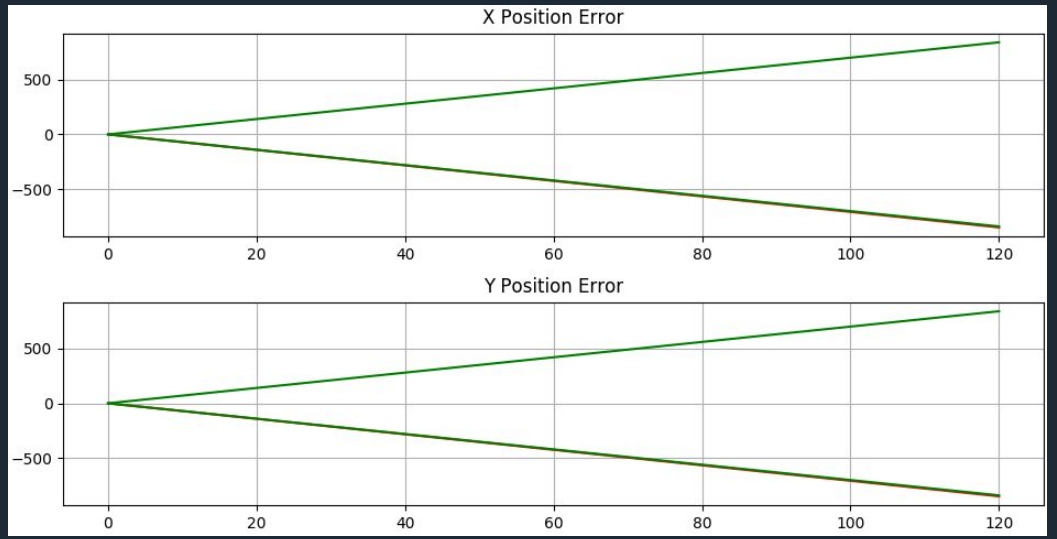
b. Run the simulation and see that the (3 Sigma) position uncertainty stays at approximately +/-15m



Assignment 1: Prediction

Step 7) Check the Velocity Covariance Prediction is Working Correctly

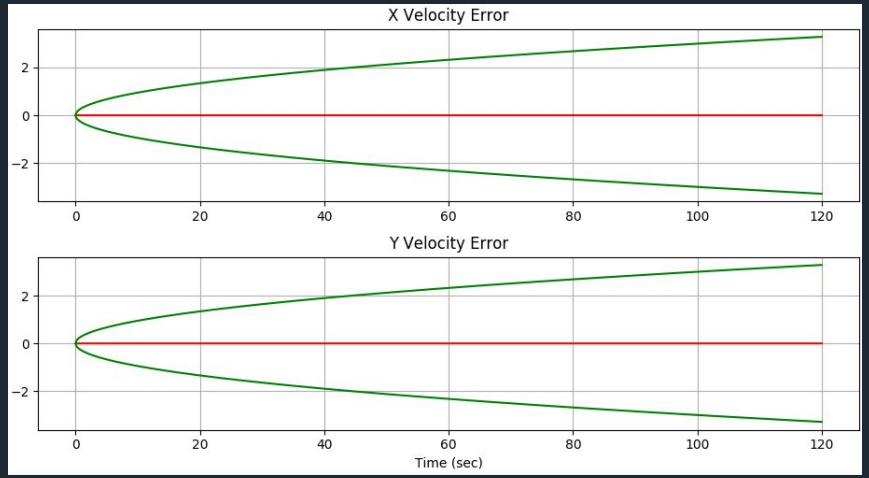
a. Set the initial state to be all zero
b. Set the initial position covariance to zero and the initial velocity x and y covariance to be (7/3)^2
c. Run the simulation and see that the (3 Sigma) error position uncertainty grows at the same rate as the position changes



Assignment 1: Prediction

Step 8) Check the Acceleration Covariance Prediction is Working Correctly

a. Set the initial state back to the original value
b. Set the initial covariance to be all zero
c. Set the process model accel_std to be 0.1
d. Run the simulation and see that the (3 Sigma) velocity uncertainty grows quadratically with time



2D Tracker Example: Prediction Step Explanation

Steven Dumble, PhD

Filter State Prediction Model Check

- Good Practice:
  - If you have a simulation of the model you are trying to estimate, then run the simulation without any noise from a **known initial condition**, and run the Kalman Filter prediction from the **same initial condition** as see that the two are consistent.
  - Run without update step, so that errors in the model aren't hidden

Filter Covariance Prediction Model Check

- The covariance must always represent the approximate level of uncertainty in the filter estimates (Model and Tuning need to be correct)
  - *True State is within the Uncertainty Bounds around the Estimated State*
  - If this is not the case then the filter is **Inconsistent**. This leads to bad and incorrect estimates.

Filter Covariance Prediction Model Check

- Check prediction model correctly transforms uncertainty
  - Uncertainty in the system should not be increasing (if Q = 0), but it can be transformed with model

Transforms Uncertainty from *k-1* to *k*

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

Inflates/Increases Uncertainty

Filter Covariance Prediction Model Check

- Check Position Uncertainty operating as expected
  - Set initial position uncertainty to non-zero value
  - Set initial velocity uncertainty to zero
  - Run the filter and see that the position uncertainty stays the same

Filter Covariance Prediction Model Check

- Check Velocity Uncertainty operating as expected
  - Set initial position uncertainty to zero
  - Set initial velocity uncertainty to non-zero value
  - Run the filter and see that the velocity uncertainty stays the same, but the position uncertainty grows linearly with time.
    - 1 m/s (1 sigma) velocity uncertainty will grow into a 120 m (1 sigma) uncertainty after 120 seconds

Filter Covariance Prediction Model Check

- Check Process Model Noise (Acceleration Uncertainty) operating as expected
  - Set initial position uncertainty to zero
  - Set initial velocity uncertainty to zero
  - Set non-zero acceleration std (So Q Process Model Noise is non Zero)
  - Run the filter and see that the total level of uncertainty in the system grows with time (velocity uncertainty increases linearly, position uncertainty increases quadratically)

Process Model Noise

- Process Model Noise increases the uncertainty in the system to capture the stochastic nature
  - Simple case here does not need increased uncertainty as the true model has no random input (true velocity is constant, no acceleration, physics is the same as the kalman filter model)
  - Increasing the process model noise can compensate for a kalman filter process model that does not correctly model the true system (as it causes the filter to stay **consistent**)

Steven Dumble, PhD

Kalman Filter: Update Step

Steven Dumble, PhD

Update Step (State)

Measurement Model: $z_k = \mathbf{H}_k x_k + \mathbf{M}_k v_k$

- We now need to derive how to update the state **prediction** $\hat{x}_k^-$ with the current measurement z_k to form the **updated** state estimate $\hat{x}_k^+$
- This will be done as a weighted estimate based on the **error** between the **measurement** z_k and the **predicted measurement** calculated from $\mathbf{H}_k \hat{x}_k^-$

Innovation:

$$\tilde{y}_k = z_k - \mathbf{H}_k \hat{x}_k^-$$

Innovation Covariance:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k$$

$\mathbf{M}_k \mathbf{R}_k \mathbf{M}_k^T$

$z_k \sim$ Measurement Vector
$\hat{z}_k \sim$ Predicted Measurement Vector
$\tilde{y}_k \sim$ Measurement Innovation (Error) Vector
$\mathbf{S}_k \sim$ Innovation Covariance Matrix

Update Step (State)

- The state is updated based the size of the innovation $\hat{y}_k$, using the **Kalman Gain** matrix $\mathbf{K}_k$, which itself is formed as a ratio between the innovation uncertainty and the current state uncertainty.

State Update:

$$\hat{x}_k^+ = \hat{x}_k^- + \mathbf{K}_k \tilde{y}_k$$

Kalman Gain:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{S}_k^{-1}$$

Recursive Least Squares:

$$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k \hat{x}_{k-1})$$

$$K_k = P_{k-1} H_k^T (H_k P_{k-1} H_k^T + R_k)^{-1}$$

Update Step (Covariance)

- The whole point of the update process is to reduce the uncertainty in the estimates and improve the accuracy, so the covariance of the estimate must change.

Covariance Update:

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_k^-$$

Update Step

Updated (*a posteriori*) state estimate:

$$\hat{x}_k^+ = \hat{x}_k^- + \mathbf{K}_k \left(z_k - \mathbf{H}_k \hat{x}_k^- \right)$$

Updated (*a posteriori*) covariance estimate:

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \mathbf{P}_k^-$$

Kalman Gain:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{S}_k^{-1}$$

Innovation Covariance:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k$$

Linear Kalman Filter Equations

System Model:

$$x_k = \mathbf{F}_{k-1} x_{k-1} + \mathbf{G}_{k-1} u_{k-1} + w_{k-1}$$
$$z_k = \mathbf{H}_k x_k + v_k$$

Assumptions:

$$w_k \sim N(0, \mathbf{Q}_k)$$
$$v_k \sim N(0, \mathbf{R}_k)$$

Gaussian Distribution with **zero mean** and given **covariance matrix**

$$E(w_k w_j^T) = \mathbf{Q}_k \delta_{k-j}$$
$$E(v_k v_j^T) = \mathbf{R}_k \delta_{k-j}$$

Not correlated with time

$$E(w_k v_j^T) = 0$$

Process noise and Measurement noise are independant

Predict:

$$\hat{x}_k^- = \mathbf{F}_{k-1} \hat{x}_{k-1}^+ + \mathbf{G}_{k-1} u_{k-1}$$
$$\mathbf{P}_k^- = \mathbf{F}_{k-1} \mathbf{P}_{k-1}^+ \mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$$

Update:

$$\tilde{y}_k = z_k - \mathbf{H}_k \hat{x}_k^-$$
$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k$$
$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{S}_k^{-1}$$
$$\hat{x}_k^+ = \hat{x}_k^- + \mathbf{K}_k \tilde{y}_k$$
$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$$

Steven Dumble, PhD

2D Tracker Example: Update Step

Steven Dumble, PhD

2D Tracking Measurement Model

Measurement Model: $z_k = \mathbf{H}x_k + v_k$

$$v_k \sim N(0, \mathbf{R})$$

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

$$\sigma_{meas} = \sigma_x = \sigma_y$$

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix}_k = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_k$$

Assignment 1: Update

Implement the **Kalman Filter Update Equations** and the **2D Tracking Measurement Model.**

1. **Step 1)** Open the python file '**assignment1_update.py**'
    a. Run the simulation as is. See that the object starts at the origin (px,py) = (0,0) but it moves in a random direction and speed, so (vx,vy) are unknown. Each simulation run movies with a different initial velocity.

Assignment 1: Update

Step 2) Setup the H Matrix and R Matrix

a. Assume the position measurement std is 10

```
# Setup the Model H Matrix
#self.H =

# Set the R Matrix
meas_std = 10.0
#self.R =
```

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

$$R = \begin{bmatrix} \sigma_{meas}^2 & 0 \\ 0 & \sigma_{meas}^2 \end{bmatrix}$$

Assignment 1: Update

Step 3) Implement the Kalman Filter Update Step Equations

```python
def update_step(self, measurement):

    # Make Sure Filter is Initialised
    if self.state is not None and self.covariance is not None:
        x = self.state
        P = self.covariance

        # Calculate Kalman Filter Update
        z = np.array([measurement[0],measurement[1]])

        # Predicted Measurement: z_hat = H * x
        # z_hat =
```

$$\tilde{y}_k = z_k - \mathbf{H}_k \hat{x}_k^-$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{S}_k^{-1}$$

$$\hat{x}_k^+ = \hat{x}_k^- + \mathbf{K}_k \tilde{y}_k$$

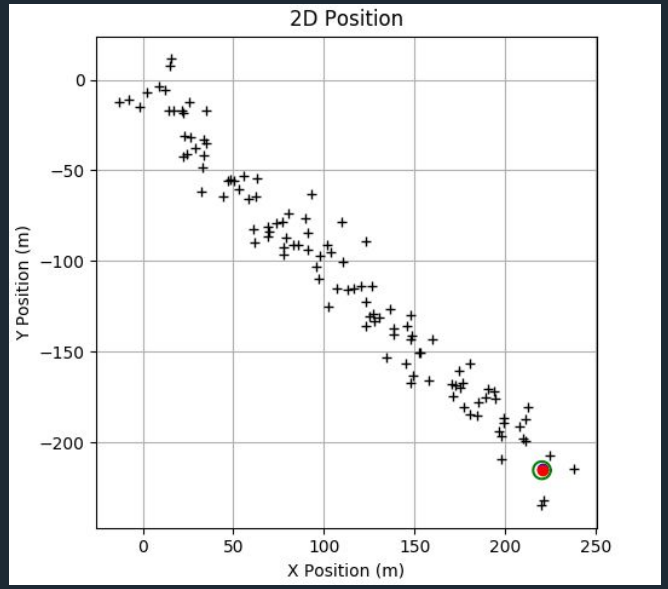
$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$$

Assignment 1: Update

Step 4) Run the Simulation

a. See that the Kalman Filter Estimate does not change or be updated. (Initial Uncertainty is zero!)

b. Change the initial velocity std to 10 and re-run the simulation.




```
Position Mean Squared Error: 13.598641307721596 (m)^2
Velocity Mean Squared Error: 1.0297190032573558 (m/s)^2
```

Steven Dumble, PhD

2D Tracker Example: Update Step Explanation

Steven Dumble, PhD

Filter Covariance Profile

Predict:

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

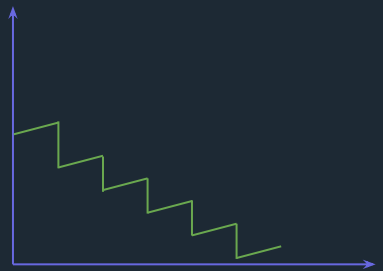
Update:

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{S}_k^{-1}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\mathbf{P}_k^-$$

Large Q, Large R / Slow Update



Small Q, Small R / Fast Update

Steven Dumble, PhD

2D Tracker Example: Initial Conditions

Steven Dumble, PhD

Assignment 1: Initial Conditions

Implement filter state and covariance initialisation on **first measurement**.

Step 1) Open the python file '**assignment1_initial_conditions.py**'

a. Run the simulation as is. See that the object does not start at the origin (px,py) = (0,0) but it starts at a random location and moves in a random direction and speed. What does the kalman filter response look like?

Assignment 1: Initial Conditions

Step 2) Set a Non-Zero Initial Covariance for the Position

a. Re-run the simulation and notice the new response

```
def initialise(self, time_step):

    # Set Initial State and Covariance (COMMENT OUT FOR DELAYED)
    init_pos_std = 0
    init_vel_std = 10
    self.state = np.array([0,0,0,0])
    self.covariance = np.diag(np.array([init_pos_std*init_pos_std,
                                        init_pos_std*init_pos_std,
                                        init_vel_std*init_vel_std,
                                        init_vel_std*init_vel_std]))
```

$$P_0 = \begin{bmatrix} \sigma_{pos}^2 & 0 & 0 & 0 \\ 0 & \sigma_{pos}^2 & 0 & 0 \\ 0 & 0 & \sigma_{vel}^2 & 0 \\ 0 & 0 & 0 & \sigma_{vel}^2 \end{bmatrix}$$

Assignment 1: Initial Conditions

Initialize at t=0:

$$x_0 = [0, 0, 0, 0]^T$$

$$P_0 = \begin{bmatrix} \sigma_{pos}^2 & 0 & 0 & 0 \\ 0 & \sigma_{pos}^2 & 0 & 0 \\ 0 & 0 & \sigma_{vel}^2 & 0 \\ 0 & 0 & 0 & \sigma_{vel}^2 \end{bmatrix}$$

Initialize on first update:

$$x_0 = \begin{bmatrix} z_0 \\ 0 \\ 0 \end{bmatrix}^T$$

$$P_0 = \begin{bmatrix} \mathbf{R} & \vdots & \vdots \\ \dots & \sigma_{vel}^2 & 0 \\ \dots & 0 & \sigma_{vel}^2 \end{bmatrix}$$

Assignment 1: Initial Conditions

Step 3) Use delayed initialisation, initialise filter on first measurement

a. Comment out the filter initialisation in the initialize function
b. Initialise the filter state and covariance on the first update
c. Re-run the simulation and notice the new response

```python
def update_step(self, measurement):

    # Make Sure Filter is Initialised
    if self.state is not None and self.covariance is not None:
        x = self.state
        P = self.covariance
        H = self.H
        R = self.R

        # Calculate Kalman Filter Update
        z = np.array([measurement[0],measurement[1]])
        z_hat = np.matmul(H, x)

        y = z - z_hat
        S = np.matmul(H,np.matmul(P,np.transpose(H))) + R
        K = np.matmul(P,np.matmul(np.transpose(H),np.linalg.inv(S)))

        x_update = x + np.matmul(K, y)
        P_update = np.matmul( (np.eye(4) - np.matmul(K,H)), P)

        # Save Updated State
        self.innovation = y
        self.innovation_covariance = S
        self.state = x_update
        self.covariance = P_update

    else:

        # Set Initial State and Covariance
        return
```

Steven Dumble, PhD

Technitute_

Kalman Filter: Tuning and Performance

Steven Dumble, PhD

Kalman Filter Validation

How can we tell if the Filter is Working or Tuned Correctly?

- Compare Kalman Filter State Estimates with the True States
  - Use a simulation to obtain the truth for comparison
  - Use an external reference system

State Estimation Error

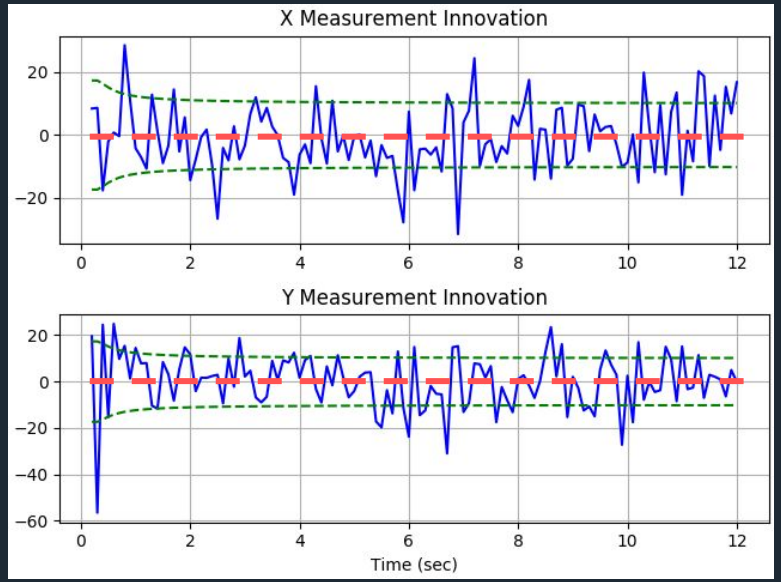
$$\tilde{x} = x - \hat{x}$$

State Estimation Covariance

$$P = [(x - \hat{x})(x - \hat{x})^T]$$

- Compare Innovation Statistical Properties
  - Zero Mean
  - Innovation Sequence Variance should be similar to the Innovation Covariance

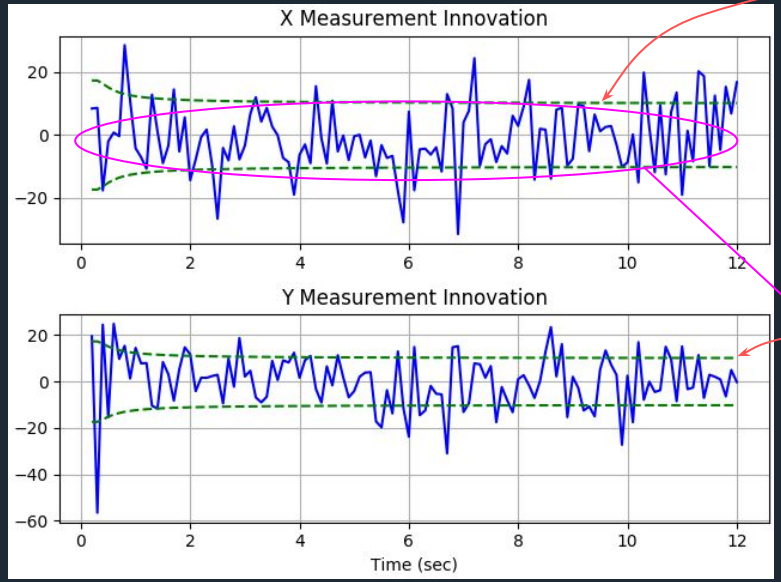
Innovation Validation (Mean)



Zero Mean

Zero Mean

Innovation Validation (Variance)



$$S_i = \begin{bmatrix} \sigma_x^2 & \sigma_x \sigma_y \\ \sigma_y \sigma_y & \sigma_y^2 \end{bmatrix}$$

$$\sigma_x^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i)^2$$

```
X Position Measurement Innovation Std: 11.332088180105783 (m)
Y Position Measurement Innovation Std: 10.330967400479567 (m)
```

Kalman Filter Tuning

How can we tune the filter?

Initial Conditions:

$\hat{x}_0$ ~ Best guess or assumption about the initial state

P_0 ~ Variance which is large enough to encompass the truth

Covariance Matrices:

R ~ Sensor Measurement Covariance -> Properties of Sensor

Q ~ Process Model Covariance -> Small enough to allow the filter to work in all situations

$$z_1 = x_1 \pm a$$
$$z_2 = x_2 \pm b$$
$$\Downarrow$$
$$\sigma_{z_1} = 3a$$
$$\sigma_{z_2} = 3b$$
$$\Downarrow$$
$$R = \begin{bmatrix} \sigma_{z_1}^2 & 0 \\ 0 & \sigma_{z_2}^2 \end{bmatrix}$$

Kalman Filter Performance

Correctly Tuned and Consistent filter, the performance can be judged based on:

- Overall Estimation Error for a series of runs / typical performance
- Size of the state covariances

  - Position Measurements: 10m (1 Sigma)
  - Position Estimates 20m (1 Sigma)

  Useless, better just to use the raw measurements

  - Position Measurements: 10m (1 Sigma)
  - Position Estimates 3m (1 Sigma)

  Good, better position estimates from filter than raw measurements

Steven Dumble, PhD

Technitute_

Kalman Filter: Implementation Notes

Steven Dumble, PhD

Numerical Accuracy

- The equation for the Covariance Matrix after the update assumes that the computational implementation has perfect numerical accurate. Numerical inaccuracies can cause stability issues.

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_k^-$$

- The *Joseph Stabilized* version is more stable and robust:

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T$$

Divergence Issues

Theoretically the Kalman Filter is an attractive choice for estimation and data fusion. However when the Kalman Filter is implemented in real systems it may not work, even though the theory and equations are correct.

Issues caused by:

- Finite precision (computation limits)
- Modelling errors (incorrect system modelling/matrices/assumption)

Divergence Issues

You can use several strategies if you encounter these issues:

1. Increase arithmetic precision (i.e. use 64 bit numbers, instead of 32 bit)
2. Use a form of the square root kalman filter
3. Symmetrize the covariance matrix at each time step
4. Initialize the covariance matrix as best as possible, to stop large jumps in matrix (i.e. don’t initialise P with large numbers)
5. Use a form of fading-memory filter
6. Increase the process model noise (Q matrix) to account for modelling errors)

Steven Dumble, PhD

Kalman Filter: Summary

Steven Dumble, PhD

Linear Kalman Filter Equations

System Model:

$$x_k = \mathbf{F}_{k-1} x_{k-1} + \mathbf{G}_{k-1} u_{k-1} + w_{k-1}$$
$$z_k = \mathbf{H}_k x_k + v_k$$

Assumptions:

$$w_k \sim N(0, \mathbf{Q}_k)$$
$$v_k \sim N(0, \mathbf{R}_k)$$

Gaussian Distribution with **zero mean** and given **covariance matrix**

$$E(w_k w_j^T) = \mathbf{Q}_k \delta_{k-j}$$
$$E(v_k v_j^T) = \mathbf{R}_k \delta_{k-j}$$

Not correlated with time

$$E(w_k v_j^T) = 0$$

Process noise and Measurement noise are independant

Predict:

$$\hat{x}_k^- = \mathbf{F}_{k-1} \hat{x}_{k-1}^+ + \mathbf{G}_{k-1} u_{k-1}$$
$$\mathbf{P}_k^- = \mathbf{F}_{k-1} \mathbf{P}_{k-1}^+ \mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$$

Update:

$$\tilde{y}_k = z_k - \mathbf{H}_k \hat{x}_k^-$$
$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k$$
$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{S}_k^{-1}$$
$$\hat{x}_k^+ = \hat{x}_k^- + \mathbf{K}_k \tilde{y}_k$$
$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$$

Linear Kalman Filter Tuning and Validation

Validation:

Minimise Error Between Kalman Filter State Estimates and the True States

$$\min RMSE$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\tilde{x}_i)^2}$$

Tuning:

Ensure Measurement Innovation is close to Zero Mean

$$\bar{z} \approx 0$$

$$\bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i$$

Ensure Measurement Innovation Variance is close to the Actual Innovation Variance

$$\bar{S} \approx \sigma_z^2$$

$$\sigma_z^2 = \frac{1}{N}\sum_{i=1}^{N}(z_i)^2$$

Steven Dumble, PhD

Technitute_

Linear Kalman Filter Conclusion

Introduction Series

Steven Dumble, PhD

Course Outcomes

We have covered:

- How to probabilistically express uncertainty using probability distributions
- How to convert differential systems into a state space representation
- How to simulate and describe state space dynamic systems
- How to use Least Squares Estimation to solve estimation problems
- How to use the Linear Kalman Filter to solve optimal estimation problems
- How to derive the system matrices for the Kalman Filter in general for any problem
- How to optimally tune the Linear Kalman Filter for best performance
- How to implement the Linear Kalman Filter in Python

Conclusion

- This should be **everything you need to know** to get you started with the linear kalman filter for sensor fusion and state estimation. You should be able to **handle problems** or situation that arise with the fundamental knowledge in this course.
- The reference documents and cheat sheets included are a handy reference.
- You definitely now on your way to becoming a **Subject Matter Expert** (SME) on all things Kalman Filter!

Conclusion

- Thanks for taking the time to finish the course!
- I hope you have enjoyed the course and learnt a lot
- Please consider my other courses for your future learning adventures

The End

Thanks for your time and feedback!

Please let me know what you liked about the course and what can be improved.

Steven Dumble, PhD