Robot_Control

Yinchheanyun

3 July 2023

What is a dynamical system?

A **dynamical** system is a system whose behavior changes over time, often in response to external stimulation or forcing.

What is a control system?

A control system is an interconnection of two or more dynamical systems that provides a desired response.

- Control is to modify the inputs to the plant (system) to produce a desired output.
- Modern control systems include physical and cyber components.
- A **physical component** is a mechanical, electrical, fluid, or thermal device acting as a sensor, or actuator.
  - A **sensor** is a device that provides measurements of a signal of interest (output)
  - An **Actuator** is a device that alters the configuration of the system or its environment (input)
- A **cyber component** is a software node that executes a specific function
- **Control system engineering** focuses on:
  - **Modeling** cyber-physical systems;
  - **Analyzing** the system behavior;
  - **Designing** controllers that achieve desired system performance characteristics;

Why automatic control?

Two broad categories of control

- Manual Control
- Automatic Control

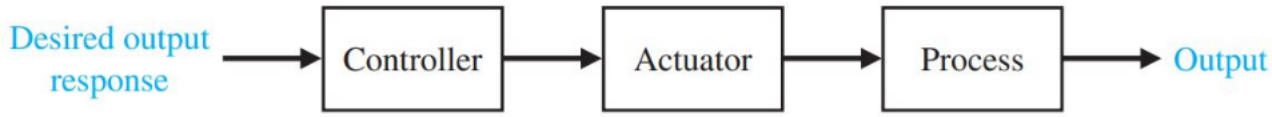
Reasons for using automatic control:

- Reduce workload and Perform tasks people can't
- Reduce the effects of disturbances and plant variations
- Stabilize an unstable system

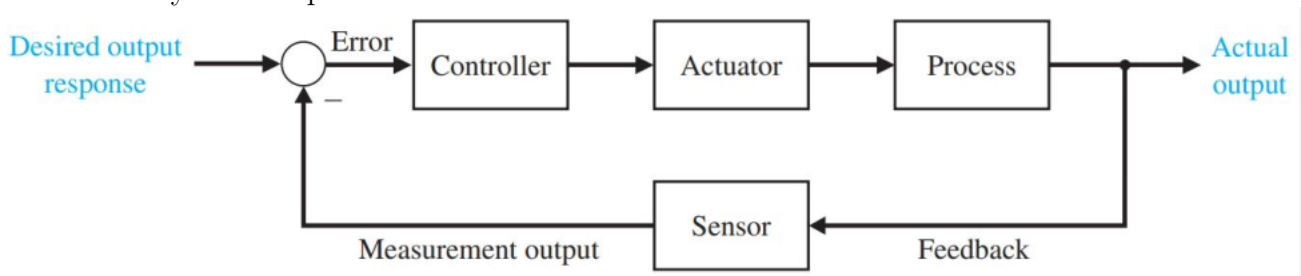
- Improve the performance of a system (time response)
- improve the linearity of the system

Open-loop vs Closed-loop Control Systems?

- An **open-loop** (Feedforward) control system utilizes a controller without measurement feedback of the system output.



- A **closed-loop** (Feedback) control system utilizes a controller with measurement feedback of the system output.



I INPUT

Input or Desired output response such as Voltage,Velocity,Trajectory,Torque,force,Wind,parameter or Variable and etc.

1. **Velocity**

   1. 1 **why you use Velocity ?**
      Velocity can use to find Modeling of Motor .

   1. 2 **How to find Velocity of DC Motor when you have Voltage?**

      - you most have a Encoder_Motor to measure velocity(omega(rd/s))
      - formula to find Velocity motor(rd/s)
        we have voltage
        - **Step1:Generate voltage to PWM**
          you can Generate voltage to PWM thought Microcontroller(Simulink_MATLAB,Arduin
          If In STM32 you can use (TIMER Generate PWM)
        - **Step2:Calculate speed Motor omega(rd/s)**
          i. find count per revolution motor thought read_encoder
          ii. you can follow formular below:
          _$dcount = count_old - count_new$.
          _dt you can choose between from 10ms to 20ms.
          _$V_{count} = dcount/dt$.
          _$V_{(count/min)} = V_{count}/1000$.
          _$V_{(revolution/min)} = V_{count/min}/(Countperrevolution)$
          _$Omega_{rd/s} = 2 * pi * V_{(revolution/min)}$

2. **Frequency_Control**

2.1 What is Frequency?

Frequency is the rate at which current changes direction per second. It is measured in hertz (Hz), an international unit of measure where 1 hertz is equal to 1 cycle per second. Hertz (Hz) = One hertz is equal to one cycle per second. Cycle = One complete wave of alternating current or voltage.

2.2 Type of Frequency

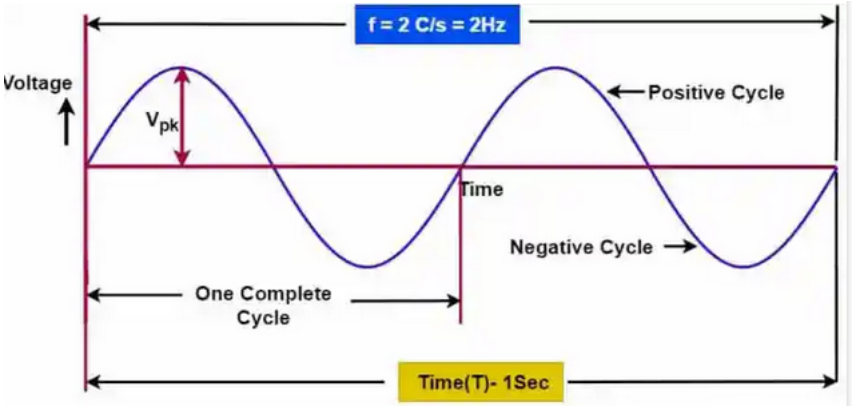
2.21 Rotational frequency:

_Rotational frequency also known as rotational speed or rate of rotation (symbols v, lowercase Greek nu, and also n), is the frequency of rotation of an object around an axis.

_ we can convert angular frequency to rotational frequency by $v = \frac{\omega}{2\phi}$

where :

- v is rotational frequency, with unit cycles per second
- ω is angular frequency, with unit radian per second or degree per second
  For example, a stepper motor might turn exactly one complete revolution each second. Its angular frequency is 360 degrees per second (360/s), or 2ϕradians per second (2ϕ rad/s), while the rotational frequency is 60 rpm.



2.22 Angular frequency

angular frequency (symbol), also called angular speed and angular rate, is a scalar measure of the angle rate (the angle per unit time) or the temporal rate of change of the phase argument of a sinusoidal waveform or sine function (for example, in oscillations and waves).

$$y(t) = sin(\delta(t)) = sin(\omega t) = sin(2\phi f t)$$

2.3 Formula Frequency

- The frequency formula in terms of time is given as:

$$\mathbf{F = \frac{1}{T}}$$

where:

- f is the frequency in hertz measured in m/s.
- T is the time to complete one cycle in seconds.

Example 1: Using the frequency formula, find the frequency of a wave where one cycle is completed in 0.5s.
To find: Frequency
Given:Time = 0.5s
=> $f = \frac{1}{T} = \frac{1}{0.5}$=0.2 .

Example 2: Determine the frequency of the pendulum that takes 4 seconds to complete one cycle.
Given: T=4s.
$=> F = \frac{1}{4} = 0.25$

- The frequency formula in terms of wavelength and wave speed is given as,

$$\mathbf{F} = \frac{\mathbf{v}}{\lambda}$$

Where:

- v is the wave speed in m/s, and
- λ is the wavelength of the wave in m

Example: Find the frequency of lightwave when the wavelength of the light is 600nm.
To find: Frequency
Given: Wavelength $= 600nm = 60010^{-9}m = 610^{-7}m$
We know that the speed of the light $= 310^8$ m/s
$=> F = \frac{v}{\lambda} = \frac{3\times10^8}{6\times10^{-7}} = 5 \times 10^{14} sec^{-1}$

3. **Trajectory & Position_Contro**

- Bezier_path
- Cubic_Spline
- 8_Shape

4. **Torque**

4.1 **What is Torque**

Torque is the measure of the force that can cause an object to rotate about an axis. Force is what causes an object to accelerate in linear kinematics. Similarly, torque is what causes an angular acceleration. Hence, torque can be defined as the rotational equivalent of linear force. The straight line about which the object rotates is called the axis of rotation. In physics, torque is simply the tendency of a force to turn or twist. Different terminologies such as moment or moment of force are interchangeably used to describe torque. The distance of the point of application of force from the axis of rotation is sometimes called the moment arm or lever arm.

4.2 **Types of Torque**

The two types of torque are static and dynamic, discussed as follows:

- **Static Torque :** is torque that does not result in an angular acceleration.When someone pushes on a closed door, the door receives a static torque because, despite the exerted force, it is not spinning about its hinges. Because they are not accelerating, someone riding a bicycle at a steady speed is also creating a static torque. Some other examples of static torque include tightening of the bolt with the help of a wrench, opening the caps of bottles using a bottle opener, turning the steering wheel of the vehicle, etc.
- **Dynamic Torque :** The torque that results in angular acceleration is called dynamic torque. When a racing car accelerates off the line, the drive shaft must be creating an angular acceleration of the wheels given that the vehicle is moving quickly around the track. Also, when you are riding a bicycle, you start pedalling and the bicycle starts to move at varying speeds, which is also an example of dynamic torque. Some more examples of dynamic torque include Spinning a top, operations of a wind turbine, and use of a power drill.

4.3 Applications of Torque

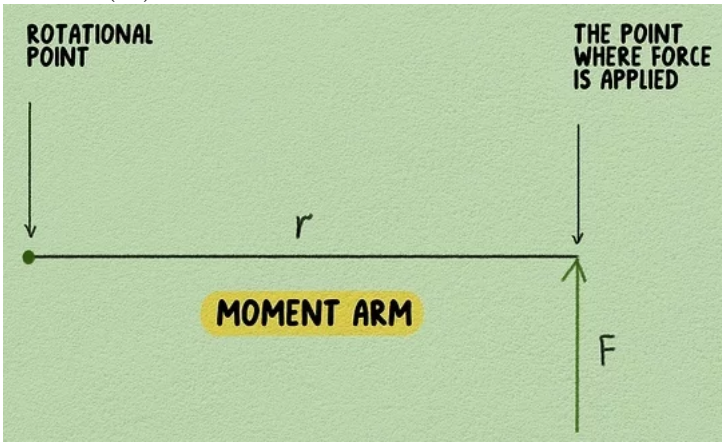
For Torque to be applied in any system, the system must have a pivot point. These are some applications of torque:

- **Automotive Industry:** In automobile industries, torque is equivalent to the measurement of the power of the engine. It is used to market a vehicle's power side by side with the horsepower of the engine. In automobiles, the engine produces torque which is further transmitted by transmission to the wheel and makes the vehicle go forward and backward.
- **Construction:** In construction, many equipment and technique are the results of the application of torque. The most common application of torque in construction can be seen in the tightening of the screw and bolts. There is a equipment called torque wrench which can apply a very specific amount of torque to a bolt or screw, ensuring precise tightening.
- **Sports:**Torque is used by many players in various games such as golf, baseball, and tennis. For example, in gold, the player uses the torque generated by the rotation of the body of the golfer to generate more power for the shot and transmit that power to the ball using the gold club.
- **Robotics:** In Robotics, torque is used for the movement of the arms and other joints of the body of the robot. The motor placed in the various parts of the robot generates torque. which allow the robot to move very precisely and perform various task.

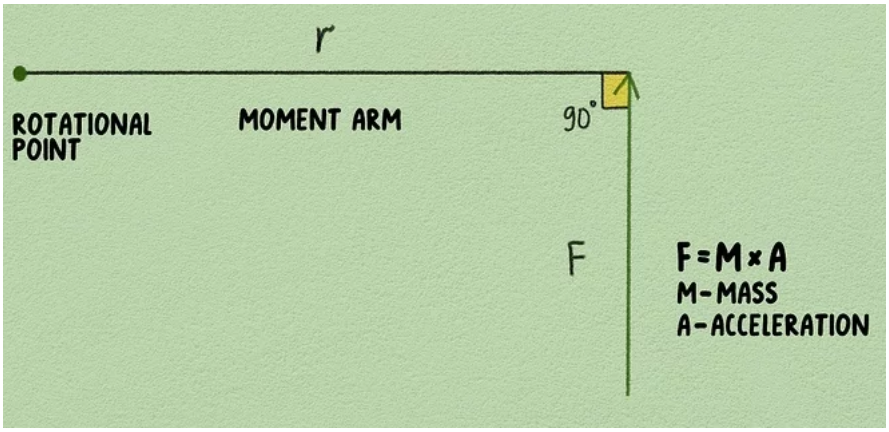
4.4 How to calculate Torque

4.4.1 Method1:Finding Torque for Perpendicular Forces

- Step1:Find the length of the moment arm.
  The distance from the axis or rotational point to the point where force is applied is called the moment arm. This distance is typically expressed in meters (m).



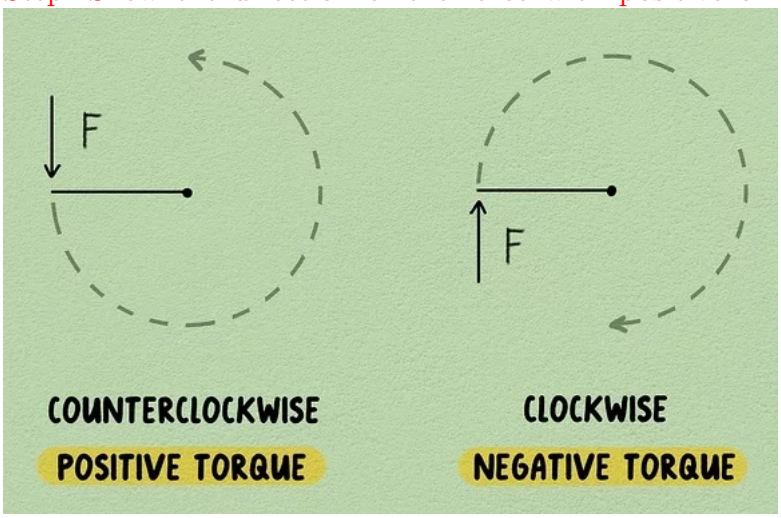
- Step2:Work out the force being applied perpendicular to the moment arm.
  The force applied perpendicular to the moment arm produces the greatest torque. The simplest torque equation assumes the force is being applied perpendicular to the moment arm.
  In torque problems, you'll typically be given the magnitude force. However, if you have to work it out yourself, you'll need to know the mass of the object and the acceleration of the object in m/s^2. According to Newton's Second Law, force is equal to mass times acceleration ($\mathbf{F} = \mathbf{m} \times \mathbf{a}$).



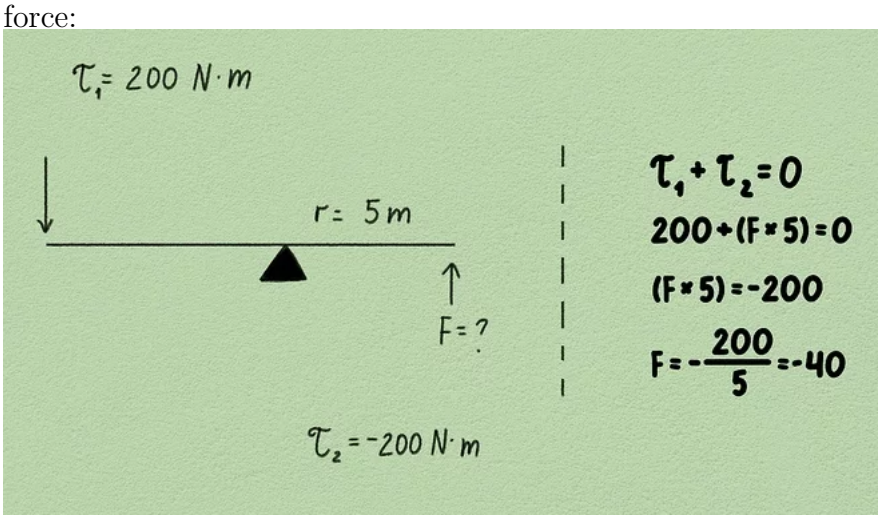
- Step3:Multiply the force times the distance to find the torque.
  The basic formula for torque is

$$\boldsymbol{\tau} = \mathbf{F} \times \mathbf{r}$$

- Step4:Show the direction of the force with positive or negative torque.

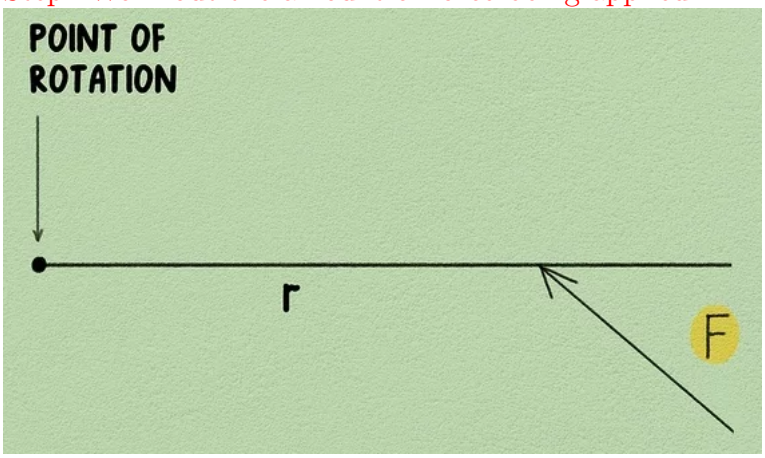


- Step5:Total individual torques around a given axis to find the net torque ($\sum \tau$).
  For example, suppose you're told that the net torque is zero. The magnitude of the torque on one side of the axis is 200 Nm. On the other side of the axis, force is being exerted from the axis in the opposite direction 5 meters from the axis. Since you know that net torque is 0, you know that the 2 forces must add up to 0, so you can construct your equation to find the missing force:

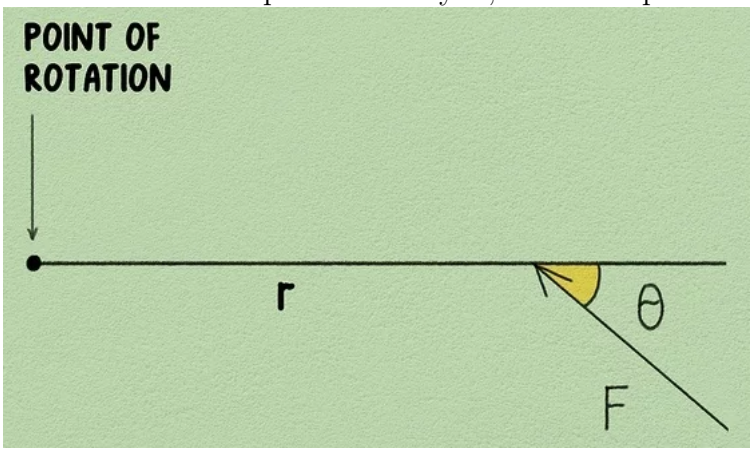


4.4.2 Method2:Figuring out the Torque for Angled Forces

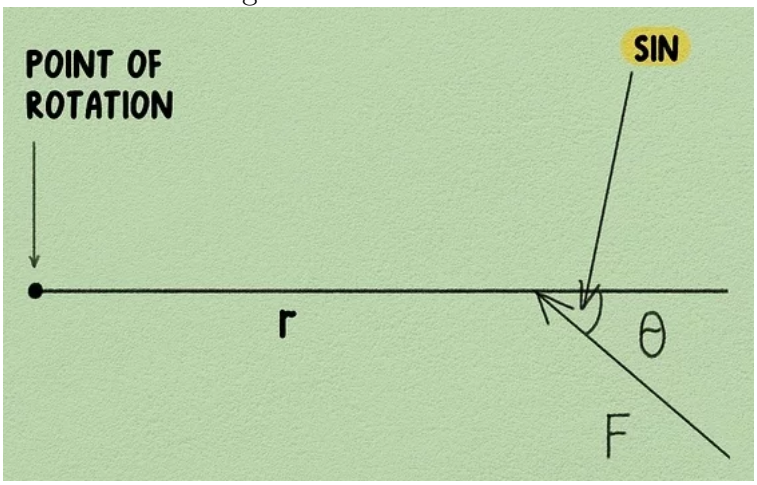
- Step1:Work out the amount of force being applied.



- Step2:Measure the angle made by the force vector and the radial vector.
The angle you measure is the one to the right of the force vector. If the measurement isn't provided for you, use a compass to measure the angle.



- Step3:Use your calculator to find the sine of the angle θ.
In the torque equation, you multiply the distance of the radial vector and the amount of force with the sine of the angle you just measured. Put the angle measurement into your calculator, then press the "sin" button to get the sine of the angle.



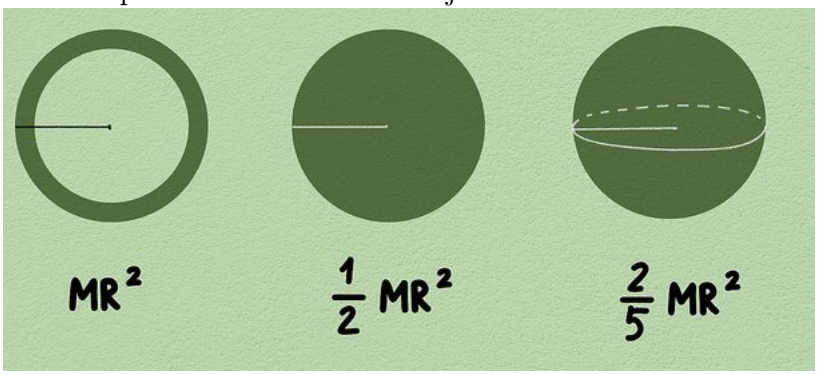
- Step4:Multiply the distance, force, and sine to find the torque.

$$\boldsymbol{\tau} = \mathbf{F} \times \mathbf{r} \times \mathbf{sin}\delta$$

For example, suppose you have a radial vector 10 meters long. You're told that 20 Newtons of force is being applied to that radial vector at a 70 angle. You would find that the torque is 188 Nm: $\tau = \mathbf{10} \times \mathbf{20} \times \mathbf{sin70} = \mathbf{10} \times \mathbf{20} \times \mathbf{0.94} = \mathbf{18}$

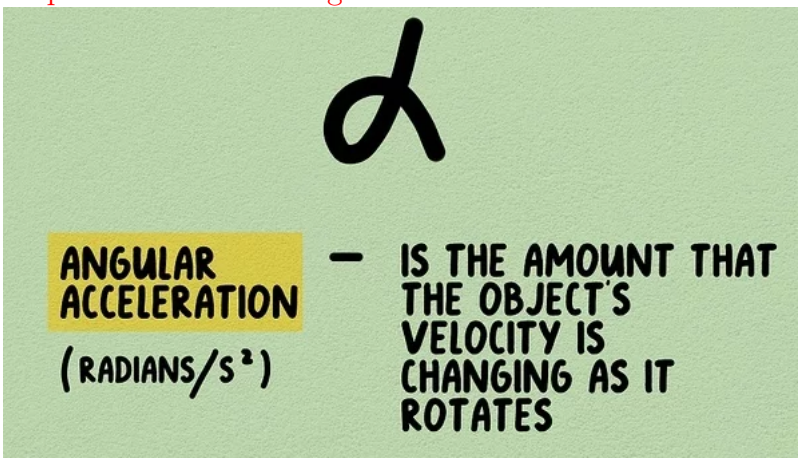
4.4.3 Method3:Determining Torque with Moment of Inertia and Angular Acceleration

- Step1:Find the moment of inertia.
  The amount of torque required to move an object with angular acceleration depends on the distribution of the object's mass, or its moment of inertia, expressed in kgm^2. When the moment of inertia isn't provided, you can also look it up online for common objects.

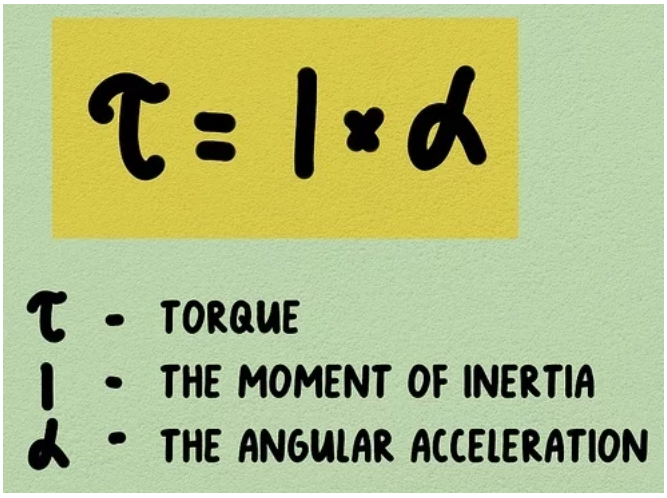


  For example, suppose you're trying to figure out the magnitude of torque on a solid disc. The moment of inertia for a solid disc is $\frac{1}{2}MR^2$. The "M" in this equation stands for the mass of the disc, while the "R" stands for the radius. If you know that the mass of the disc is 5 kg and the radius 2 meters, you can determine that the moment of inertia is 10 kgm^2.

- Step2:Determine the angular acceleration.



- Step3:Multiply the moment of inertia by the angular acceleration to find the torque.



_For example, suppose you know that the moment of inertia for an object is 10 kgm^2. You're also told that the torque is 20 Nm, but you need to find out the angular acceleration. Since you know that $\tau = I\alpha$, you also know that $\alpha = \frac{\tau}{I}$. When you put in the variables you know, you'll find that the angular acceleration for the object is 2 radianss^2 : $\alpha = \frac{20}{10} = 2$

5. **Pumb**

II Control Algorithm

1. **PID(Proportional Integral Derivative)**

   1.1 **What is PID Controller?**
   A PID controller is an instrument used in industrial control applications to regulate temperature, flow, pressure, speed and other process variables.

   1.2 **Response System**

Response of 2nd Order Systems to Step Input (0 < ζ < 1)

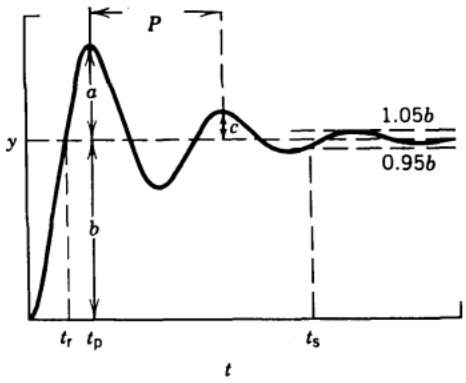


Figure 5.10. Performance characteristics for the step response of an underdamped process.

1. **Rise Time:** t_r is the time the process output takes to first reach the new steady-state value.
2. **Time to First Peak:** t_p is the time required for the output to reach its first maximum value.
3. **Settling Time**: t_s is defined as the time required for the process output to reach and remain inside a band whose width is equal to ±5% of the total change in y. The term 95% response time sometimes is used to refer to this case. Also, values of ±1% sometimes are used.
4. **Overshoot:** $OS = a/b$ (% overshoot is $100a/b$).
5. **Decay Ratio**: $DR = c/a$ (where c is the height of the second peak).
6. **Period of Oscillation**: P is the time between two successive peaks or two successive valleys of the response.

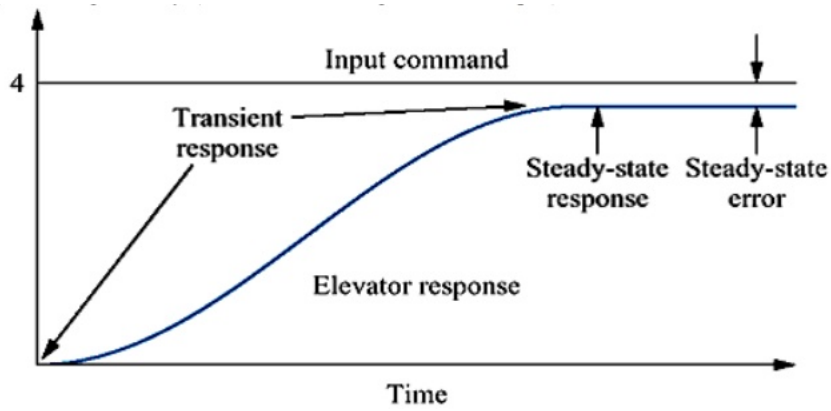
Eq. 5-51

$$y(t) = KM\left\{1 - e^{-\zeta t/\tau}\left[\cos\left(\frac{\sqrt{1-\zeta^2}}{\tau}t\right) + \frac{\zeta}{\sqrt{1-\zeta^2}}\sin\left(\frac{\sqrt{1-\zeta^2}}{\tau}t\right)\right]\right\}$$

1.3 Type error of Response System

There are two types error

- Transient error
  - When the input is applied, there is some delay in the reaction of the plant to the input due to the reluctance.
  - it overshoots and oscillations at the output are produced.
- Steady state error



1.4 Proportional

- The proportional term, denoted with P, is based on the current error between the setpoint and the measured output of the system.
- The proportional helps bring the output of the system back to the setpoint by applying a correction that is proportional to the amplitude of the error, leading to a reduction of the rise time of the correction signal.

- A Proportional controller is used to reduce the rise time and speed up the response. This controller makes no changes in the phase response of the plant.

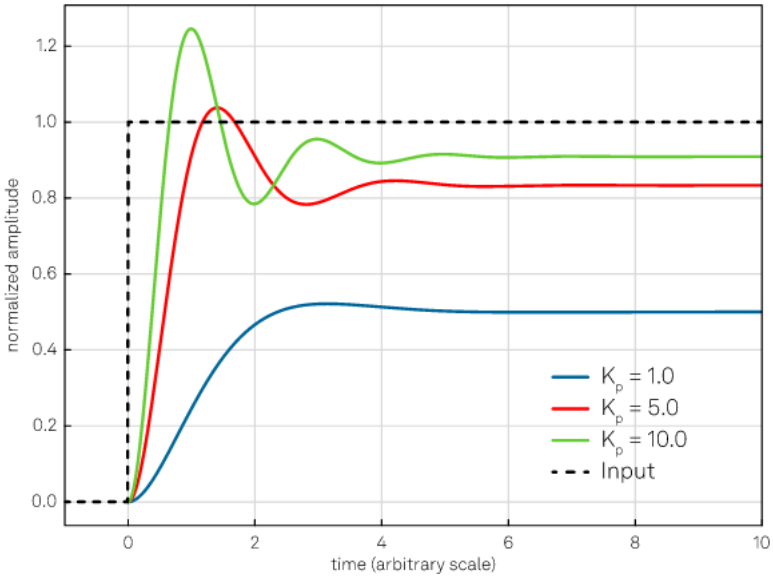


fig:Effect of the proportional action. Increasing the Kp coefficient reduces the rise time, but the error never approaches zero. Additionally, a too high value of the proportional gain might lead to an oscillating output.

1.5 Integral

- The integral term, denoted with I, applies a correction that is proportional to the time integral of the error, i.e. the history of the error. For example, if the error persists over time, the integral term continues to increase, resulting in a larger correction applied to the output of the system.
- Increasing the value of the integral gain coefficient increases the contribution of the accumulated error over time to the control signal.
- if there is a steady-state error, an integral term with a large gain coefficient will drive the control signal to eliminate the error faster than a smaller integral term.
- However, increasing the integral term too much can lead to an oscillating output if too much error is accumulated, causing the control signal to overshoot and create oscillations around the setpoint. This phenomenon is sometimes called integral windup.
- An Integral controller corrects the time invariant errors. This provides a phase lag and no change in magnitude in the output

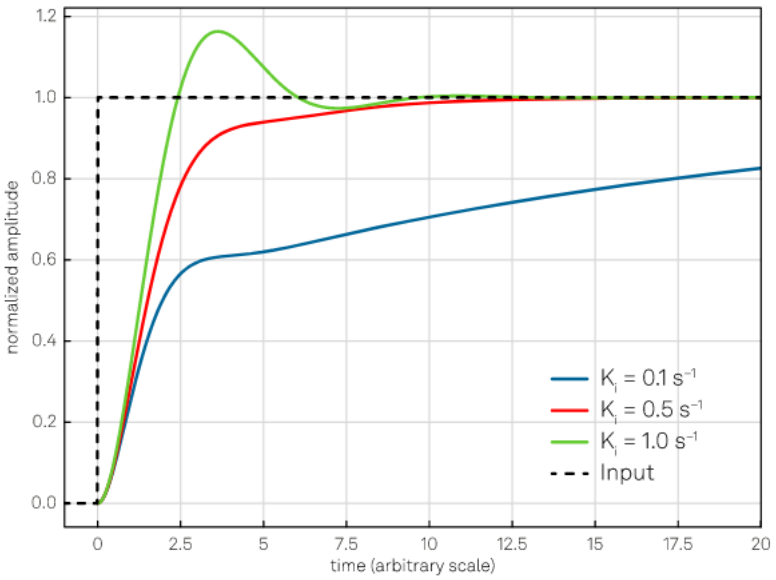
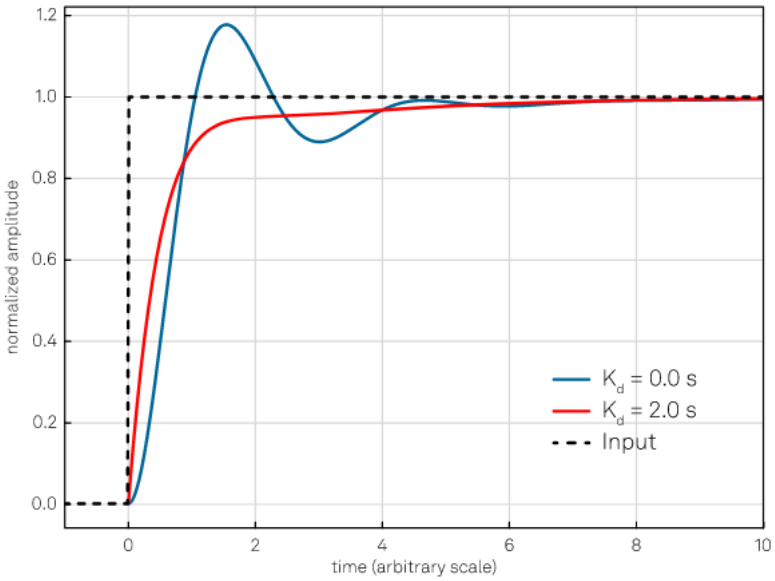


Figure:Effect of the integral action with constant Kp = 1. Increasing Ki, the response will be faster but also lead to larger oscillations and overshoot if the value increases too much (green curve).

1.6 Derivative

- The derivative term, denoted with D, provides a control over the error tendency, i.e. its future behavior, by applying a correction proportional to the time derivative of the error.
- The aim is to anticipate the changes in the error signal: if the error shows an upward trend, the derivative action tries to compensate without waiting for the error to become significant (proportional action) or for it to persist for some time (integral action).
- A Derivative controller is required to minimise the transient errors like overshoot and oscillations in the output of the plant. But this can create heavy instability in noisy environments.Be careful to use smaller gain with this controller. It provides a phase lead to the output when compared with the input. , usually with no change in magnitude.



1.7 PD Controller

A PD controller reduces the transients like rise time, overshoot, and oscillations

in the output. Useful for changing magnitude and want to add phase lead to the output.

1.8 PI Controller

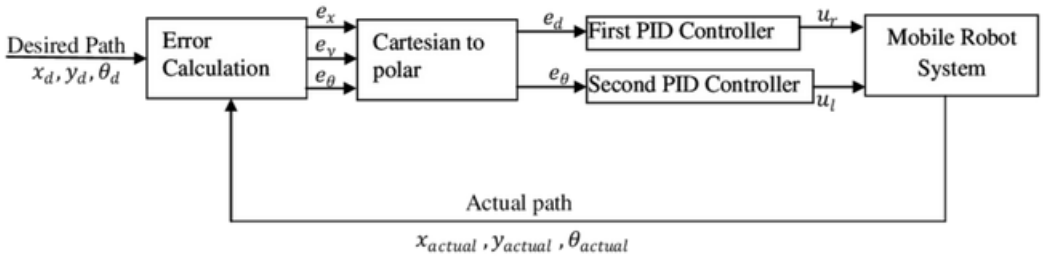
A PI controller helps in reducing both the rise time and the steady state errors of the system. To be useful whenever you need to change magnitude and lag the phase together.

1.9 PID Controller

A PID controller is a general form of controller. The gains of the three control actions can be adjusted to achieve any controller. The change in magnitude along with either lead or lag in phase in the output can be made available through this general model of controller.

1.10 How to calculate PID?

- **Step1:Calculate Error**
  Error=Reference-Measurement.
- **Step2:Calculate P,I,D**
  $Propotional = kp * Error[-1]$
  $Integral = Integral_old + ki * (Error[-1] + Error[-2]) * dt$
  $Derivative = kd * (Error[-1] - Error[-2])/dt$
  where kp,ki,kd (you can tunning on Modelling thought MATLAB_Simulink or tunning by hand(observe)
- **Step3:Calculate Output**
  Output=Propotional + Integral + Derivative.
  This is a Diagram of PID Trajectory Tracking Controller:



This is a Diagram of PID Control DC_Motor:

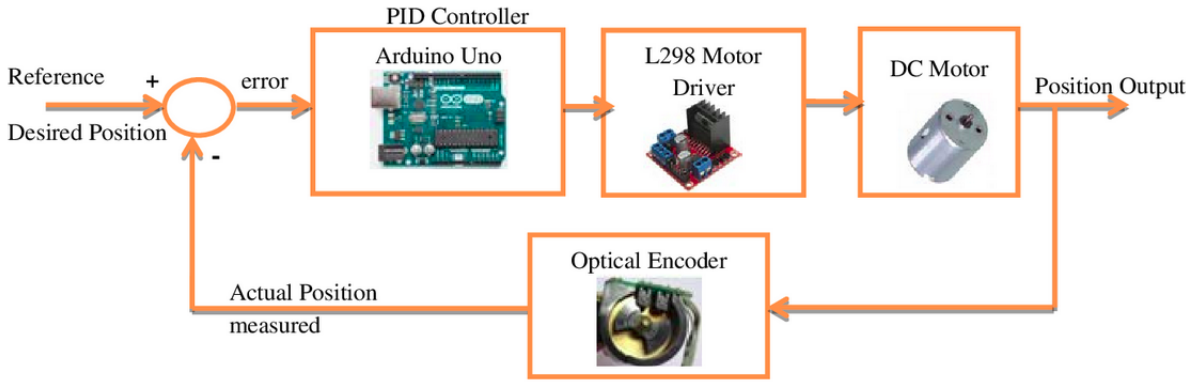


Figure 1: Block Diagram for DC Motor Position Control

1.11 Modifications to the algorithm

1.11.1 Integral windup

What is Integral Windup?

Integrator windup or reset windup, refers to the situation in a PID controller where a large change in setpoint occurs (say a positive change) and the integral

term accumulates a significant error during the rise (windup), thus overshooting and continuing to increase as this accumulated error is unwound (offset by errors in the other direction). The specific problem is the excess overshooting.

Why use Integral Windup

- **Overshooting and Oscillations**: Integral windup can lead to significant overshooting of the setpoint when the control output is saturated. This can result in poor control performance, especially in systems where precise and stable control is required.
- **Instability:** In some cases, integral windup can lead to control system instability, causing the controlled variable to become unmanageable and potentially damaging the system.
- **Actuator Wear and Tear:** When the control output remains at its maximum or minimum limits for extended periods due to integral windup, it can cause unnecessary wear and tear on the actuator or control device, reducing its lifespan.

How to use Integral_windup?

First : Set Integral_Min, Integral_Max, Output_Min, Output_Max.
Integral winup affect on Integral of PID.
The Standard Integral= Integal_pev + ki*erro[-1]*dt.
When you use Integral winup:

```
if (Integral >= Integral_Max)
{
.  Integral=Integral_Max;
}
else if (Integral <= Integral_Min)

.  Integral=Integral_Min;

Output=Proportional + Integral + Derivative
if (controlOutput > OUTPUT_MAX)
{
.  controlOutput = OUTPUT_MAX;
}
else if (controlOutput < OUTPUT_MIN)
{
.  controlOutput = OUTPUT_MIN;
}
```

1.11.2 Deadband

What is Deadband?

Many PID loops control a mechanical device (for example, a valve). Mechanical maintenance can be a major cost and wear leads to control degradation in the form of either stiction or backlash in the mechanical response to an input signal. The rate of mechanical wear is mainly a function of how often a device is activated to make a change. Where wear is a significant concern, the PID loop may have an output deadband to reduce the frequency of activation of the output (valve). This is accomplished by modifying the controller to hold its output steady if the change would be small (within the defined deadband range). The calculated output must leave the deadband before the actual output will change.

Why use deadband in PID?

- **Noise Filtering:** In many real-world applications, the process variable (PV) being controlled can exhibit small, high-frequency variations or noise. These variations can be caused by sensor inaccuracies, environmental factors, or other sources of interference. Deadband helps filter out these noise components by preventing the control system from responding to minor fluctuations within the deadband range. This noise filtering is essential for maintaining a stable and accurate control system.
- **Reducing Control Effort:** In situations where the process variable is close to the setpoint and oscillates slightly around it, continuously adjusting the control output can be inefficient and lead to unnecessary wear and tear on control devices like control valves or actuators. Deadband allows the control output to remain constant within the deadband range, reducing the overall control effort and prolonging the life of the control device.
- **Preventing Chattering:** Chattering is the rapid and frequent switching of the control output due to small changes in the process variable. Chattering can lead to mechanical wear and adversely affect system stability. Deadband helps prevent chattering by keeping the control output stable within the deadband range until the process variable crosses the deadband threshold by a certain amount.
- **Stability Improvement:** Deadband contributes to control system stability by minimizing the number of control actions required for small deviations from the setpoint. A control system that continuously adjusts the control output for small errors can become unstable or exhibit oscillations. Deadband helps maintain a stable system response and minimizes overshoot.
- **Energy Efficiency:** Reducing unnecessary control actions within the deadband range leads to energy savings. In applications where energy consumption is a concern, deadband helps optimize control while minimizing the system's energy usage.
- **Tolerance for Process Variability:** Deadband allows for a certain level of process variability without triggering control actions. This is especially useful in applications where the process variable may naturally fluctuate within a specific range, and it's unnecessary to make continuous adjustments for these variations.
- **Prolonging Equipment Life:** Deadband helps protect control devices (e.g., control valves) from constant, small adjustments, which can lead to mechanical wear and reduce the lifespan of these components. By reducing the control effort and minimizing wear and tear, deadband contributes to prolonging the life of control system equipment.

How to use Deadband in PID ?

The Standard error=setpoint - measurement.
Let Deadband=0.5
Initialize previous control output:Previous_Control = 0.0
While Ture:
if (abs(error) <= Deadband):
{
control=previous_control
}
control=P+I+D
# update the previous control input:previous_control=control.

1.11.3 Low pass Filter

What is Low pass Filter in PID?

A low-pass filter in a PID (Proportional-Integral-Derivative) controller is a component that filters out high-frequency noise from the derivative (D) term of the PID controller's output. The purpose of the low-pass filter is to smooth the derivative term and reduce its sensitivity to rapid, high-frequency changes in the error signal. By doing so, the controller can better distinguish between genuine changes in the process variable and short-term noise, resulting in more stable and accurate control.

Why use Low pass Filter in PID?

- **Noise Reduction:** One of the primary reasons for using a low-pass filter in the derivative (D) term of a PID controller is to reduce the sensitivity of the controller to high-frequency noise in the system. High-frequency noise can lead to erratic and unstable control behavior. The low-pass filter smooths out rapid changes in the error signal, allowing the controller to focus on meaningful, low-frequency variations.
- **Stability:** High-frequency noise in the derivative term can introduce instability into the control loop. It can cause the controller to react excessively to minor fluctuations in the process variable, leading to overshoot, oscillations, and instability. The low-pass filter helps prevent these issues by filtering out the noise.
- **Improved Settling Time:** By reducing noise and excessive controller responses, the low-pass filter can help the control system settle more quickly to the desired setpoint. It allows for a smoother and more controlled response to changes in the process variable.
- **Reduction of Derivative Kicks:** The derivative term in a PID controller can lead to "derivative kicks" when there are sudden changes in the error signal. These kicks can result in large, undesirable control outputs. The low-pass filter reduces the impact of these kicks by smoothing the derivative term.
- **Improved Robustness:** A low-pass filter enhances the robustness of the control system by making it less sensitive to measurement noise and disturbances. This is particularly valuable in applications where environmental or sensor noise is a concern.
- **Customizable for Specific Systems:** The parameters of the low-pass filter, such as the cutoff frequency and smoothing factor, can be customized to match the characteristics of the specific control system. This flexibility allows engineers to fine-tune the filter's performance.

How to use Low_pass filter in PID?

- **Formular:**
  error(t)=setpoint(t)- measurement(t)
  Derivative of Positional Error after low pass filter: E(t)=(1-α)*E(t-1)+$\alpha * \frac{derror(t)}{dt}$
  α Smoothing Factor (smaller for greater smoothing) (0< α <1)
- **Code**
  _Initialize and Configure Your PID Controller:Set up your PID controller by configuring the appropriate PID gains (Kp, Ki, Kd) and initializing variables, including the integral term and the low-pass filter components.
  _Design Your Low-Pass Filter:The low-pass filter, in this context, acts as a

filter on the derivative term to reduce high-frequency noise. The filter can be a simple first-order low-pass filter, and you can use an exponential moving average (EMA) or a digital filter design method to implement it. The filter equation is:
filtered_derivative = α * derivative + (1 - α) * previous_filtered_derivative
_Inside the Control Loop:
a.Calculate the error as usual:
float error = setpoint - process_variable; // Calculate the error
b.Calculate the derivative term without filtering:
float derivative = Kd * (error - previousError); // Calculate the derivative term.
c.Apply the low-pass filter to the derivative term:
float alpha = 0.1; // Adjust the smoothing factor (alpha) as needed
float filtered_derivative = alpha * derivative + (1 - alpha) * previous_filtered_derivative;
// Store the current filtered derivative for the next iteration
previous_filtered_derivative = filtered_derivative;
e.Calculate the control output by combining the proportional (P), integral (I), and filtered derivative (D) terms:
float controlOutput = Kp * error + integral + filtered_derivative;

1.11.4 Setpoint step change

What is Setpoint step chang in PID?

A setpoint step change in a PID (Proportional-Integral-Derivative) control system is a sudden and intentional adjustment of the desired target value, known as the setpoint. This change is typically made to test or demonstrate the control system's ability to respond to abrupt variations in the desired operating conditions.

A setpoint step change involves:

- **Initial Setpoint:** The PID control system initially operates with a specific setpoint value. The system has likely reached a steady state or is in the process of stabilizing at this setpoint.
- **Sudden Change:** At a predetermined time or under specific conditions, the setpoint is changed abruptly from its current value to a new, different setpoint value. This change occurs instantaneously or over a very short period.
- **Response Observation:** The control system responds to this change by adjusting the control output to bring the process variable (PV) back to the new setpoint value. The PID controller calculates the control output based on the error, which is the difference between the new setpoint and the current PV.
- **Evaluation:** The system's response to the setpoint step change is evaluated to assess various performance characteristics, such as the rise time (time taken to reach the new setpoint), settling time (time taken to stabilize around the new setpoint), overshoot (exceeding the setpoint before stabilizing), and overall control system behavior.

Why use Setpoint Step change?

- **Testing and Validation:** Setpoint step changes are used to test and validate the performance of control systems. They allow engineers and operators to assess how well the system responds to abrupt changes in the desired operating conditions. This is crucial for ensuring that the control system

operates as intended and meets specified performance criteria.

- **Disturbance Handling:** In many real-world applications, control systems need to be able to respond to disturbances or changes in the environment. By simulating disturbances through setpoint step changes, control system designers can evaluate how well the system can reject or compensate for external influences, ensuring robustness and reliability.
- **Start-Up and Shutdown Procedures:** Setpoint step changes are often employed during start-up and shutdown procedures in industrial processes. They help transition the system from one operational state to another in a controlled and predictable manner, ensuring a smooth and safe process.

How to use Setpoint Step change?

- **Define the Setpoint Values:** Determine the initial setpoint_value (the current target value) and the final setpoint value (the new target value) that you want your system to reach. These values will depend on the specific requirements of your application.
- **Choose the Transition Time:** Decide how long the transition from the initial setpoint to the final setpoint should take. The transition time is typically specified in seconds.
- **Calculate the Rate of Change:** The rate of change, often referred to as the "ramp rate," is the rate at which the setpoint value changes over time. You can calculate it as follows: //**Rate of Change = (Final Setpoint - Initial Setpoint)** / Transition Time //For example: if you want to change the setpoint from 50 to 100 over 10 seconds:
  Rate of Change = (100 - 50) / 10 = 5 units per second
- **Implement the Setpoint Change:** To actually change the setpoint value over time, you can use a loop or a timer. In each iteration of the loop or at regular time intervals, increment or decrement the setpoint value based on the calculated rate of change.
  For example, in a loop running every 1 second:
  - Initial Setpoint: 50
  - After 1 second: 55
  - After 2 seconds: 60
  - After 3 seconds: 65
  - ...
  - After 10 seconds: 100 (final setpoint)
- **Adjust PID Controller Parameters:** Depending on your specific control system and the characteristics of your process, you may need to adjust the PID controller's parameters (Kp, Ki, Kd) to achieve the desired response to the setpoint step change. Fine-tuning is often necessary to minimize overshoot and settling time while maintaining stability.
- **Monitor and Analyze:** Continuously monitor the system's response to the setpoint step change. Evaluate key performance metrics such as rise time, settling time, overshoot, and steady-state error to ensure the system meets your control objectives.
- **Additional Considerations:** Depending on the complexity of your system and your control requirements, you may need to consider factors such as integral windup, anti-windup techniques, and deadband to optimize the response to the setpoint step change.

```
// Define setpoint step change parameters
float setpointInitial = 0.0;
float setpointFinal = 100.0;
float transitionTime = 10.0;  // seconds

// Calculate the rate of change
float rateOfChange = (setpointFinal - setpointInitial) / transitionTime;

// Get the start time
uint32_t startTime = HAL_GetTick();  // Use the appropriate STM32 function

while (1) {
    // Calculate the time elapsed
    uint32_t currentTime = HAL_GetTick();
    float elapsedTime = (currentTime - startTime) / 1000.0;  // Convert to

    // Check if the transition is complete
    if (elapsedTime >= transitionTime) {
        // Set the final setpoint
        UpdatePIDController(setpointFinal);
        break;
    }

    // Calculate the new setpoint and update the PID controller
    float newSetpoint = setpointInitial + rateOfChange * elapsedTime;
    UpdatePIDController(newSetpoint);

    // Compute the PID output and control your system's actuators
    float controlOutput = ComputePIDOutput();  // Implement your PID contro
    // Control your actuators using controlOutput
}
```

```
// Update PID controller with the new setpoint
void UpdatePIDController(float newSetpoint) {
    // This is a simplified PID controller update; you may need a more comp

    // Calculate error (the difference between the new setpoint and the cur
    float error = newSetpoint - getCurrentProcessVariable(); // Implement t

    // Calculate the proportional term
    float P = Kp * error;

    // Calculate the integral term (if used)
    integral += Ki * error;

    // Calculate the derivative term (if used)
    float derivative = Kd * (error - previousError);

    // Calculate the control output
    float controlOutput = P + integral + derivative;

    // Apply the control output to your system (e.g., adjust motor speed, h
    adjustSystemOutput(controlOutput); // Implement this function

    // Store the current error for the next iteration
    previousError = error;
}
```

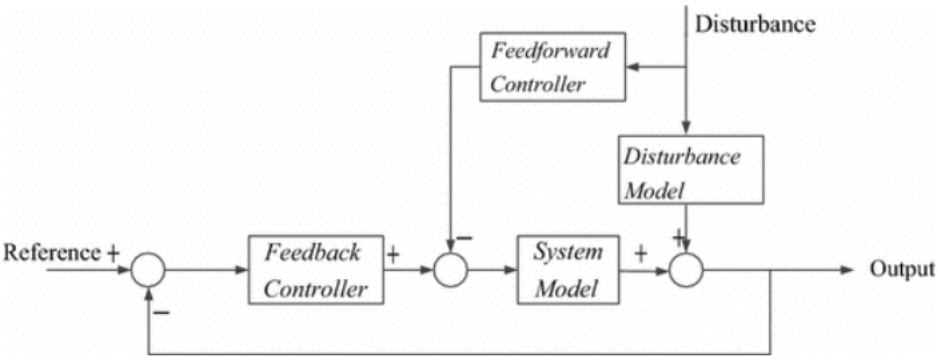
1.11.5 Feed-Forward

What is Feed_Foward?

Feed-Forward refers to a control strategy that anticipates disturbances or changes

in the system and proactively adjusts the control output to compensate for these changes. Feedforward control is typically used in combination with the traditional PID control loop to improve the overall performance of a system. It aims to reduce the reliance on feedback correction alone and provide a faster and more accurate response to changes in the system.

Block Diagram



Application:

- **Industrial Processes:**
  - **Temperature Control:** In manufacturing and industrial processes, feedforward control can be used to compensate for changes in ambient temperature, ensuring consistent product quality.
  - **Flow Control:** Feedforward control is employed to adjust flow rates based on changes in inlet pressure, fluid properties, or demand, minimizing flow variations.
  - **Load Changes:** In systems with varying loads, such as conveyor belts or material handling systems, feedforward control can anticipate load changes and adjust motor speeds accordingly.
- **Robotics:**
  - **Robotic Arm Control:** Feedforward control helps robotic arms compensate for external forces or torques acting on the arm during tasks like lifting objects.
  - **Mobile Robots:** In mobile robots, feedforward control aids in maintaining stability and trajectory tracking, even when facing uneven terrain.
- **Aerospace and Aviation:**
  - **Aircraft Control: Feedforward control is used to predict and counteract turbulence, gusts, and altitude changes, ensuring a smoother flight experience.**
  - **Spacecraft Maneuvering:** Feedforward control can be employed to anticipate and adjust for changes in spacecraft orientation due to solar radiation pressure or gravitational forces.
- **HVAC Systems:**
  - **Heating, Ventilation, and Air Conditioning (HVAC):** In HVAC systems, feedforward control can adjust heating or cooling output based on outdoor temperature and weather forecasts, leading to energy savings and maintaining indoor comfort.
- **Electric Power Systems:**
  - **Power Grid Stability:** Feedforward control can help maintain power grid stability by anticipating and managing load changes, sudden power

demands, and renewable energy fluctuations.

- **Mechatronics:**
  - **Machinery Control:** In mechatronic systems, such as CNC machines or 3D printers, feedforward control can compensate for changes in material properties or tool wear, ensuring accurate machining or printing.
- **Audio and Signal Processing:**
  - **Audio Equalization:** In audio processing, feedforward filters are used to enhance audio quality by predicting and reducing noise or distortion.
  - **Communication Systems:** Feedforward control can reduce the impact of channel noise in communication systems, leading to better data reception and transmission.

Why use Feed-Forward in PID?

Feedforward control is used in conjunction with a PID (Proportional-Integral-Derivative) controller to improve the overall control performance of a system. The primary reasons for using feedforward in PID control are as follows:

- **Improved Disturbance Rejection:** Feedforward control can proactively compensate for anticipated disturbances or changes in the system. By knowing in advance how certain changes or inputs affect the system, feedforward can minimize the impact of these disturbances, reducing the need for the PID controller to react to errors after they occur. This leads to better disturbance rejection and faster system response.
- **Faster Response Time:** Feedforward control enables the system to respond more quickly to changes in the setpoint or disturbances. Without feedforward, the PID controller would need to detect errors before taking corrective action. With feedforward, the control system can preemptively adjust the control output based on the predicted effects of changes or disturbances, resulting in faster response times.
- **Reduced Overshoot and Oscillations:** By mitigating the effects of disturbances before they impact the system's output, feedforward control helps reduce overshoot and oscillations. This is especially valuable in systems where overshooting the setpoint or system limits is undesirable, as it can lead to instability and decreased product quality.
- **Enhanced Setpoint Tracking:** Feedforward control assists in tracking a changing setpoint accurately. It can predict the required control output adjustments as the setpoint changes, allowing the system to follow the setpoint more closely without relying solely on the integral component of the PID controller.
- **Minimized Control Effort:** By reducing the need for the PID controller to continuously adjust the control output to counteract disturbances, feedforward control minimizes the control effort, which can extend the life of control equipment, reduce wear and tear, and save energy.
- **Improved System Efficiency:** In many cases, feedforward control can optimize the system's efficiency by reducing the need for excessive correction actions. This is particularly important in applications where energy efficiency is a concern, such as HVAC systems or industrial processes.
- **Stable Control in Dynamic Systems:** In systems with rapid changes or where the system dynamics vary significantly, feedforward control can help maintain stability by predicting and compensating for these dynamic changes.

How feedforward control works in a PID system:

- **Standard PID Control Loop:**
  In a typical PID control loop, the controller continuously calculates the control output based on the error, which is the difference between the desired setpoint and the actual process variable. The control output is adjusted based on this error to bring the system closer to the setpoint.
  Output=Propotional+Integral+Derivatice;
- **Disturbances and Changes:**
  Real-world systems often encounter disturbances or changes that can affect the process variable. These disturbances can be external factors like load changes, temperature fluctuations, or variations in raw materials. Changes in the setpoint, such as transitioning to a new target value, can also affect the system.
- **Feedforward Component:**
  Feedforward control introduces an additional component to the control system. This component calculates the control output adjustments required to counteract the anticipated effects of known disturbances or changes. It does this before these effects are observed in the process variable. The feedforward component doesn't rely on error correction; instead, it acts proactively.
- **Predictive Model:**
  The key to feedforward control is having a predictive model that describes how disturbances or changes influence the process variable. This model can be mathematical, empirical, or based on engineering knowledge. It quantifies the expected relationship between inputs and outputs.
- **Calculation of Feedforward Control Output:**
  When a known disturbance or change is anticipated, the feedforward component calculates the control output adjustment needed to counteract the expected impact. This calculation is typically based on the predictive model and is done independently of the PID feedback loop.
- **Combination of Feedforward and Feedback:**
  The calculated feedforward control output adjustment is combined with the output of the PID controller. The PID controller continues to operate as usual, providing feedback-based corrections to errors.
- **Real-Time Response:**
  As the system operates, the combined action of the PID feedback and feedforward components ensures that the system responds to disturbances and changes more rapidly and accurately. The feedforward component can act in advance of the PID controller detecting errors, resulting in improved control performance.
- **Feedback Correction:**
  The PID feedback component fine-tunes the control output based on real-time observations and error correction. It is responsible for handling unanticipated disturbances or errors that the feedforward component might not account for.

How to Control Feed-Forward

- **Static feedforward control:**
  If we define a generic process model(Gp), and generic disturbance model,GD,as:
  _Gp= generic CO− > PV process model (describing how a CO change will impact PV)

_GD= generic D− > PV disturbance model (describing how a D change will impact PV)

$$\textbf{COfeedforward} = -(\frac{\textbf{GD}}{\textbf{Gp}})\textbf{D}$$

- We do not account for the size of the process time constant, Tp,relative to the disturbance,TD.As a consequence,we cannot compute and deploy a series of corrective control actions over time to match how fast the disturbance event is causing the PV to move up or down.
- We do not consider the size of the process dead time, θ_p, relative to the disturbance, θ_D.Thus, we cannot delay the implementation of corrective actions to coordinate their arrival with the start of the disturbance disruption on PV.

Visualizing the action of this static feed-forward element as a two step "prediction and corrective action" procedure for a single disturbance:

- The **D− > PV** disturbance gain,KD, receives a change in D and predicts the total final impact on PV.The computation can only account for information contained in KD,which include the direction and how far PV will ultimately travel in response to the measured D before it settles out at a new steady state.
- The **CO− > PV** process gain, Kp the uses this disturbance impact prediction of "which direction and how far" to back-calculated one CO move as a corrective control element (FCE) to cause an "equal but opposite" response in PV.

COfeedforward in Normally Zero

An important implementation issue is that COfeedforward should equal zero when D is at its design level of operation (DLO) value.Thus,the D used in our calculations is actually the disturbance signal from the sensor/transmitter (Dmeasured)that has been shifted or biased by the design level of operation disturbance value (DDLO), or :

D=Dmeasured-DDLO

1.11.6 Forward PID and Backward PID

1.12 PID Control Velocity

1.13 PID Control Position

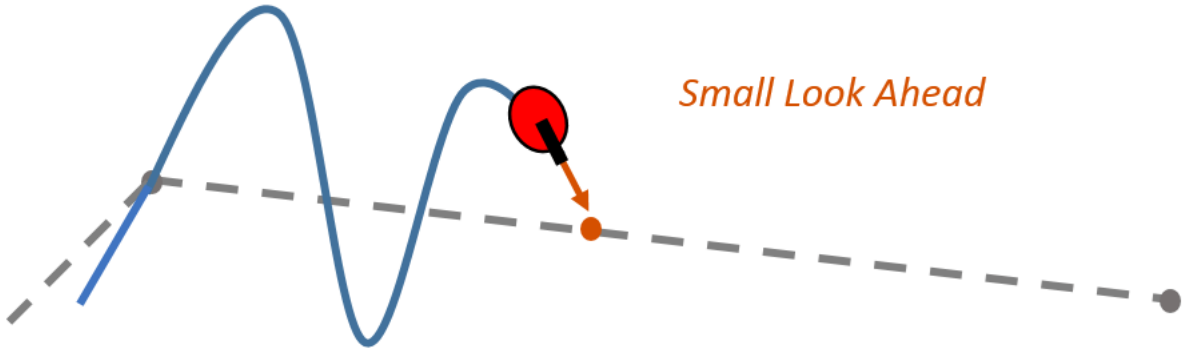
1.14 PID Control Torque

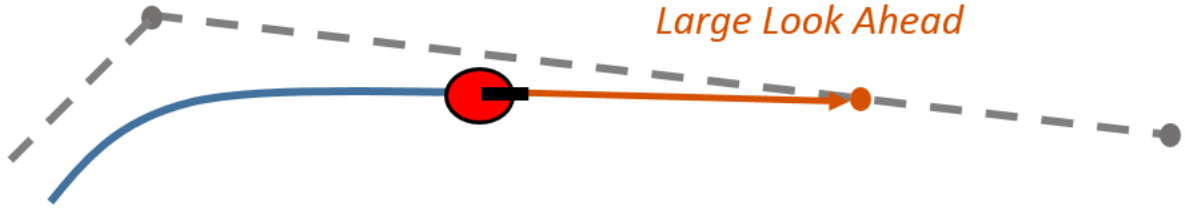
2. Pure pursuit

2. 1 What is Pure_pursuit?

Pure pursuit is a path tracking algorithm. It computes the angular velocity command that moves the robot from its current position to reach some look-ahead point in front of the robot. The linear velocity is assumed constant, hence you can change the linear velocity of the robot at any point.

_Look_Ahead_Distance=It is property is the main tuning property for the controller.





_Limitations=There are a few limitations to note about this pure pursuit algorithm:

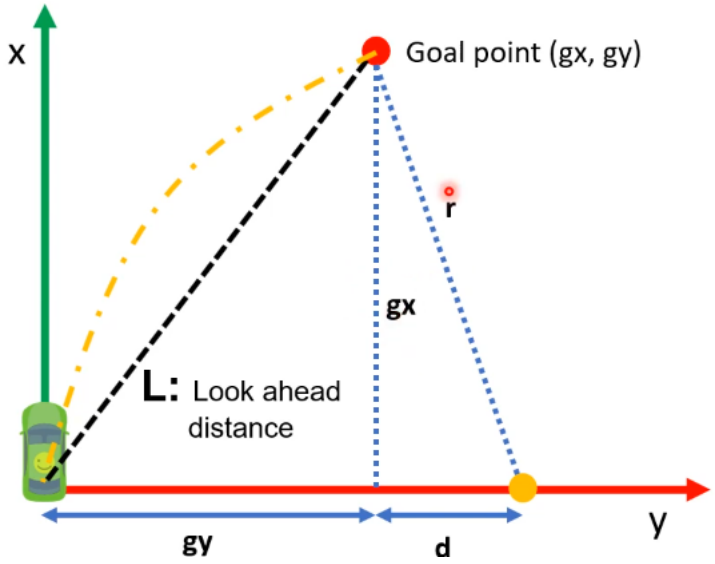
- As shown above, the controller cannot exactly follow direct paths between waypoints. Parameters must be tuned to optimize the performance and to converge to the path over time.
- This pure pursuit algorithm does not stabilize the robot at a point. In your application, a distance threshold for a goal location should be applied to stop the robot near the desired goal.

2. 2 **How does pure pursuit work?**

+ Assumption to Consider

_The path has a squence of 2D waypoints to follow.The vehicle knows that sequence.
_The Vehicle can localize it self in the given waypoint system of reference.
_The goal is to follow these waypoints.

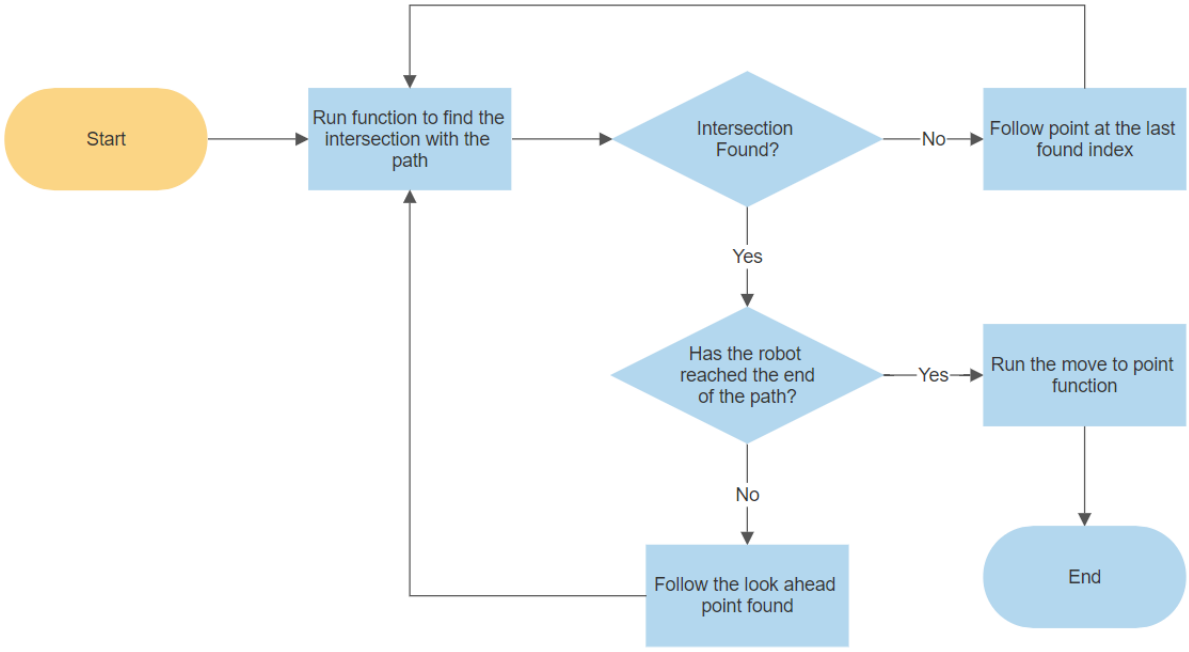
+ Geometrical interpretation



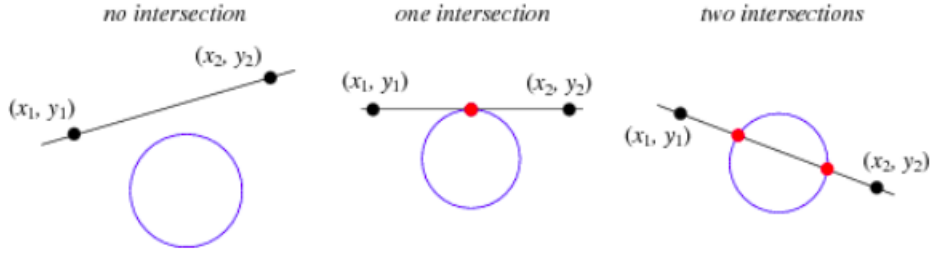
Some quick math

$$
\begin{aligned}
r &= |gy| + d \\
d^2 + gx^2 &= r^2 \\
(r - |gy|)^2 + gx^2 &= r^2 \\
r^2 + |gy|^2 - 2 * r * |gy| + gx^2 &= r^2 \\
r^2 + L^2 - 2 * r * |gy| &= r^2 \\
r &= \frac{L^2}{2 * |gy|}
\end{aligned}
$$

Steering angle should be propotional to the curlture of the Arc $\alpha = \frac{1}{r} = \frac{2|gy|}{L^2}$.
These steps will be repeated until the end of the path is reached



+Line-Circle Intersection



Ingeometry, a line meeting a circle in exactly one point is knows as a tangent line,while a line meeting a circle in exactly two points in known as a section line.
_Defining: $d_x = x_2 - x_1; d_y = y_2 - y_1; dr = \sqrt{d_x^2 + d_y^2}$

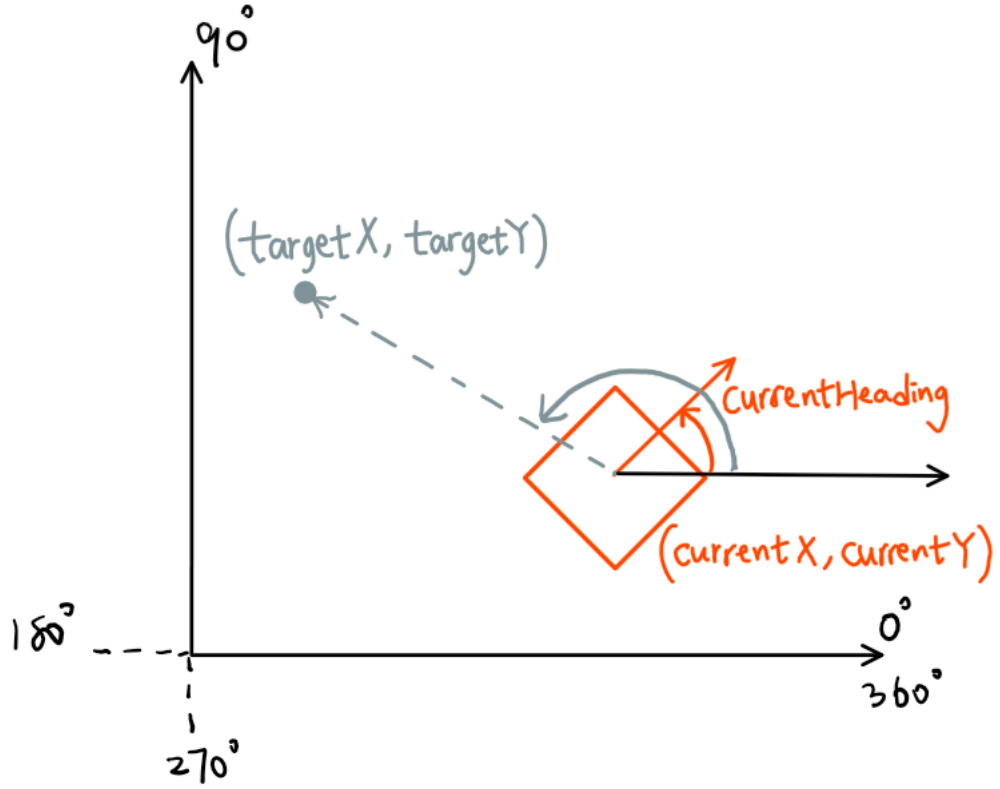
$$D = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1$$

_Gives the points of intersection as

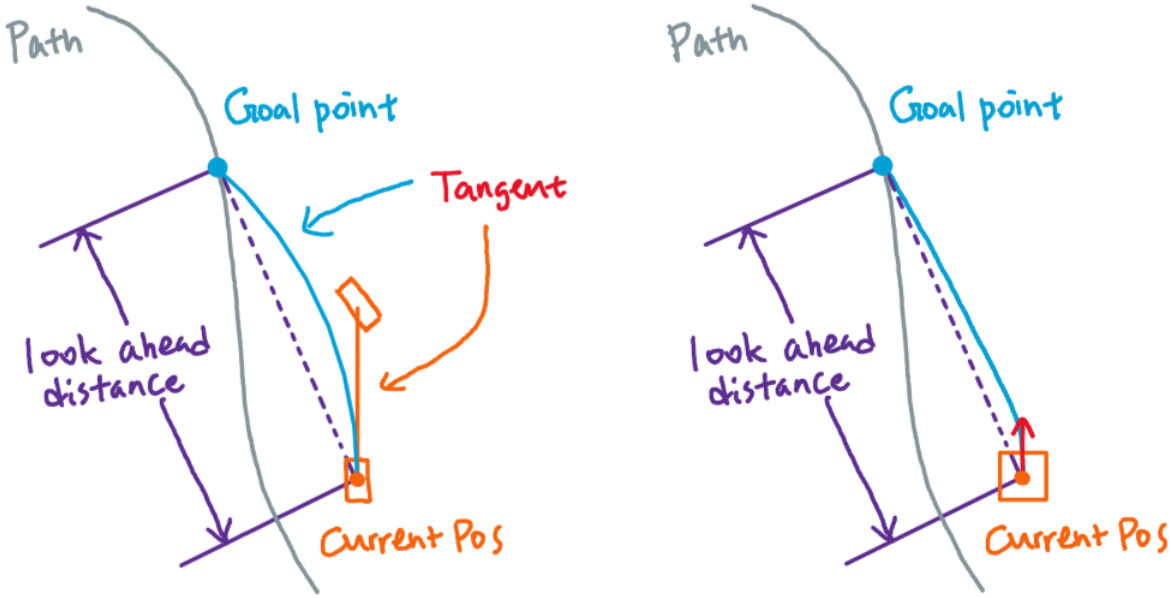
X=$\frac{Dd_y \pm sgn*(dy)dx\sqrt{r^2dr^2-D^2}}{dr^2}$
Y=$\frac{-Dd_x \pm |dy|\sqrt{r^2dr^2-D^2}}{dr^2}$
_The discriminant:$\triangle = r^2dr^2 - D^2$

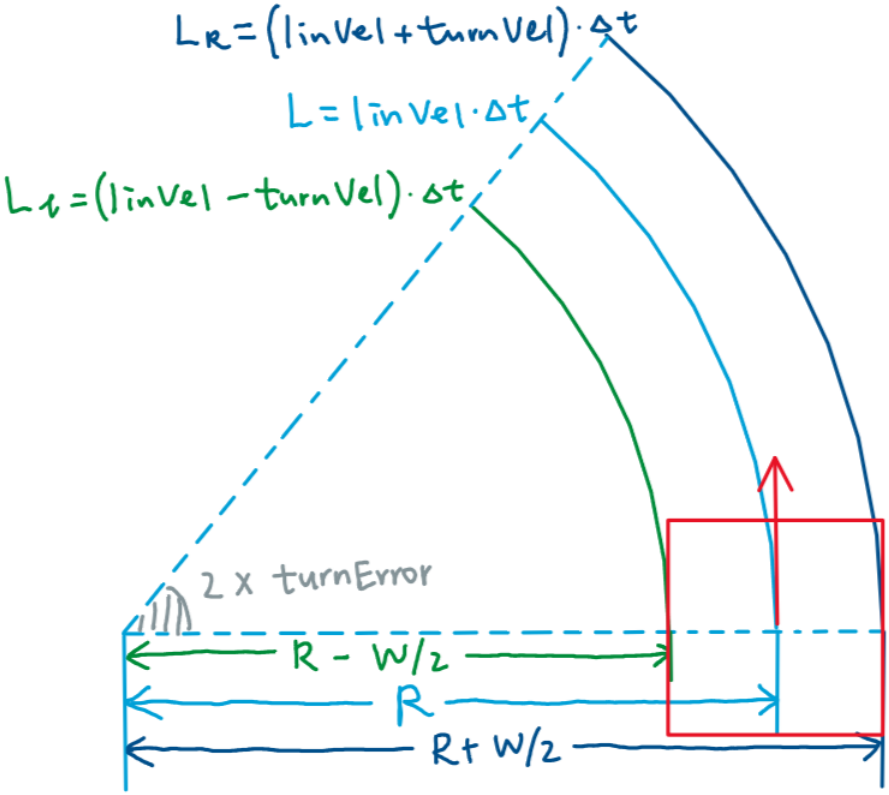
$(\triangle < 0 : nointersection; \triangle = 0 : tangent; \triangle > 0 : intersection)$

+Move Toward Target Point



_Compute linea error: linearError= $\sqrt{(targetx - currentx)^2 + (targety - curre}$
_Compute turn error:
absTargetAngle = atan2(targetY - currentY, targetX - currentX)
if absTargetAngle < 0 : absTargetAngle += 360
turnError = find_min_angle(absTargetAngle, currentHeading)
wher find_min_angle=absTargetAngle-currentHeading.
_send comment to motors
_leftside motor speed=linearVel - turnVel
_rightside motor speed=linearVel + turnVel
+Going Back to angular Velocity





$$\_R = \frac{lookAhead}{2sin(turnError)}$$
$$\_L_l = (R - \frac{W}{2})(2turnError)$$
$$\_L_r = (R + \frac{W}{2})(2turnError)$$
$$\_turnVel = \frac{W * sin(turnError) * linearVel}{lookAhead\_distance}$$

3. **Stanely Control**

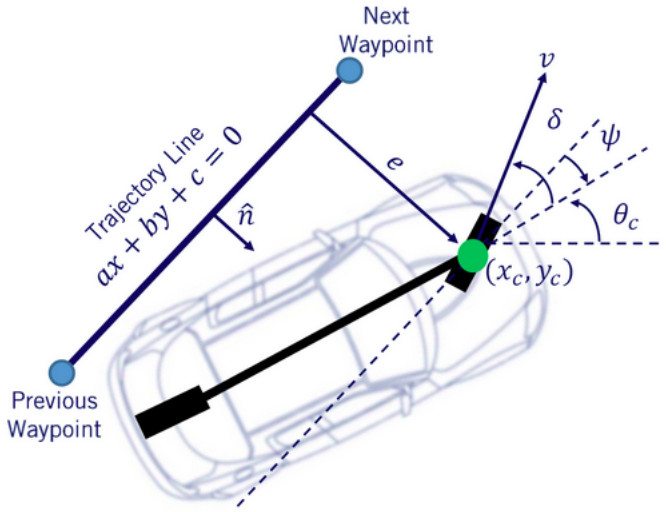
3. 1 **Stanely controller Approach**

Stanely method is the path tracking approach used by standford university's Durpa challeng team.

- Use the center of the front axle as a reference point.

- Look at both the error in heading and the error in position relative to the closet point on the path.
- Define an intuitive steering law to:
  _Correct heading error
  _Correct position error
  _obeymax steering angle bounds

3. 2 Stanley formulation



ψ is the angle between the trajectory heading and the vehicle heading. The steering angle is denoted as δ.There are three intuitive steering laws of Stanley method:

- Cross track error:
$$e = \frac{ax_c + by_c + c}{\sqrt{a^2 + b^2}}$$
- Cross track Steering:
$$\delta = tan^{-1}(\frac{Ke(t)}{V(t)})$$
- Heading error:
$$\psi = tan^{-1}(\frac{-a}{b}) - \theta_c$$
- Total steering Input:
$$\delta = \psi + tan^{-1}(\frac{Ke(t)}{V_f(t)})$$

3. 3 Case stady

- Two Scenarios
  _Large initial crosstrack error
  _Large initial heading error
- Large initial crosstrack error
  _Crosstrack error of 5 meters
  _Max Steer δ=25deg ,foreard speed of $V = 5\frac{m}{s}$
  _Gain K=2.5 ,length L=1m
  _Effect of speed Variation

4. LQR Control(Linear Quadratic Regulator)

4. 1 What is LQR Control?

- LQR is a method that calculates the optimal feedback gain K. The feedback gain can determined by tuning of Q and R. The feedback gain is used to control the system in control signal form.
- The main focus of an LQR control system is to achieve and hold a target configuration for a given linear system using an optimal control law.
- An LQR control system generates the control law using four matrices:
  - A Matrix:physical dynamics
  - B Matrix:control dynamics
  - Q Matrix:state cost
  - R Matrix:control cost

4. 2 LQR vs PID Controllers

Advantages of using LQR instead of PIDs:

- Less calibration/tuning is required
- LQR can control all axes of movement simultaneously compared to PIDs,which can only control one axis per controller
- Only one LQR controller vs. multiple PID controllers leads to simpler software architecture.

4. 3 Setting up the optimization problem

_Positive Semidefinite

$$X^T Q X >= 0$$

_Positive definite

$$U^T R U > 0$$

If Q "is bigger" than R => fast regulation of X->0,U is Large
If R "is bigger" than Q => slow regulation of X->0,U is Small

4. 4 Solving the optimization problem

We are using one of the optimal control method which is Linear Quadratic Regulator.
Minimize the cost:

$$J = \frac{1}{2}\sum_{k=0}^{N}(X_{k+1}^T Q X_{k+1} + u_k^T R u_k)$$

or

$$J = \int_0^\infty (X^T Q X + U^T R U) dt$$

4. 5 Mathematics Model

The General form of the linearized model

$$X_t = A_{t-1} X_{t-1} + B_{t-1} U_{t-1}$$

$$X_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} cos(\theta_{t-1})dt & 0 \\ sin(\theta_{t-1})dt & 0 \\ 0 & dt \end{bmatrix} \begin{bmatrix} v_{t-1} \\ \omega_{t-1} \end{bmatrix}$$

4. 6 Design Controller

_The Optimal feedback gain:

$$K = R^{-1} B^T P$$

_Algebraic Riccati Equation

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

Which

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

Where P is symmetric Require that R is invertible (logical as otherwise no weighting on input activity)

_The optimal input

$$U_t = -KX_t$$

4. 7 **State-Feedback control**

Our objective is to design a state-feedback that can give us a design poition control of the mobile robot.

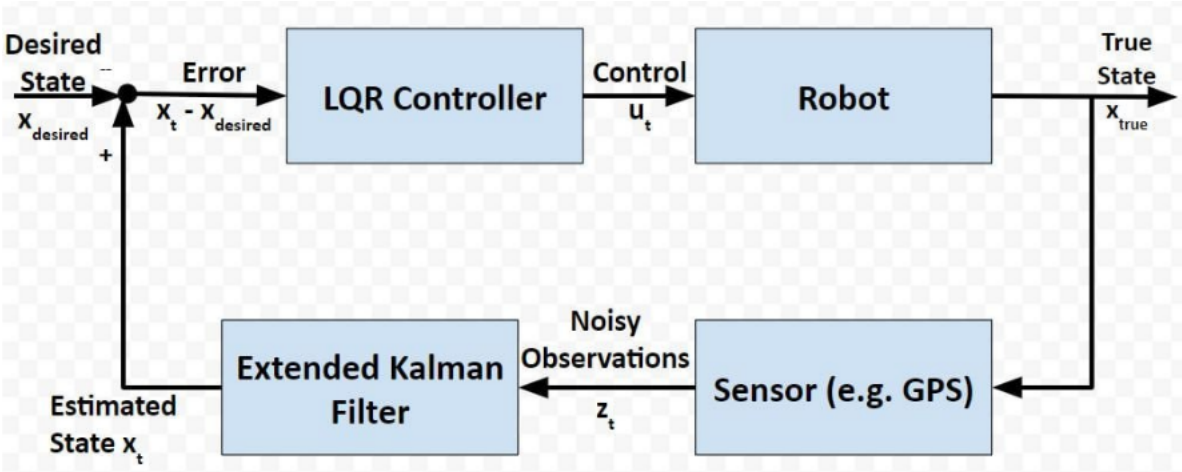
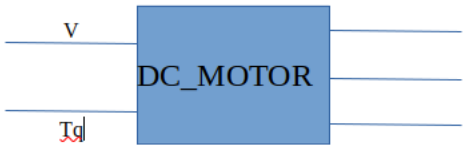


Figure: Diagram of state-feedback control

4. 8 **Step for Calculate of LQR**

Step1:Given A & B (From Plant)
Step2:Choose Q & R
Step3:Solve Algebraic Ricatti equation for P
Step4:Compute $K = R^{-1}B_T P$
Step5:Choose the K is Solution that yields stable system.

4. 9 **Example:**



$$X = \begin{bmatrix} \theta \\ \omega \\ i \end{bmatrix}; U = [Va]$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -0.29 & 71.13 \\ 0 & -63.24 & -1020.35 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 0 \\ 641.81 \end{bmatrix}$$

i. Control Saturation

ii. Inability to measure all states. use LQR to adress

5. Kalman_filter

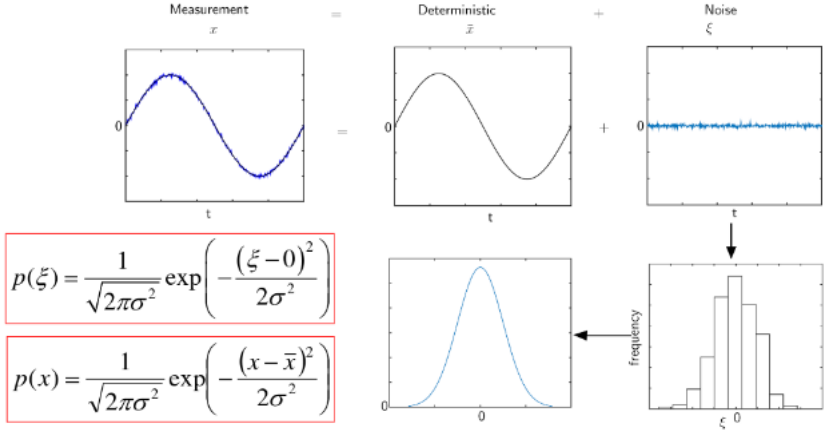
5.1 Introduction

Kalman Filter is a probabilistic states estimate technique.It can be found in:

- **Sensor Fusion:** Measurements are available from various sensors but might be subjected to noises.
- **State Estimation:**States or Variable of interest can only be measured indirectly.
- **System Identification and so on.**

Assumption

$$\mathrm{E}\left\{\begin{bmatrix} v[k] \\ w[k] \end{bmatrix}\right\} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \mathrm{E}\left\{\begin{bmatrix} v[k] \\ w[k] \end{bmatrix} \begin{bmatrix} v[k]^T & w[k]^T \end{bmatrix}\right\} = \begin{bmatrix} Q_k & 0 \\ 0 & R_k \end{bmatrix}$$



$$p(\xi) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\xi-0)^2}{2\sigma^2}\right)$$

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\bar{x})^2}{2\sigma^2}\right)$$

Kalman Filter for linea system

- **Plant:**
  _x[K+1]=A[k]x[k]+b[k]u[k]+v[k]
  _y[k]=C[k]x[k]+$\omega[k]$
- **In matrix form** $\begin{bmatrix} x[k+1] \\ y[k] \end{bmatrix} = \begin{bmatrix} A[k] & B[k] & I & 0 \\ C[k] & 0 & 0 & I \end{bmatrix} \begin{bmatrix} x[k] \\ u[k] \\ v[k] \\ \omega[k] \end{bmatrix}$
- **Take expectation of the above equation base on the observation up to [k-1] we obtain** $E\left\{\begin{bmatrix} x[k+1] \\ y[k] \end{bmatrix} |Y[k-1]\right\} = \begin{bmatrix} A[k] & B[k] & I & 0 \\ C[k] & 0 & 0 & I \end{bmatrix} E\left\{\begin{bmatrix} x[k] \\ u[k] \\ v[k] \\ \omega[k] \end{bmatrix} |Y[k-1]\right\}$

. $Ex[k+1]|Y[k-1] = x_{k+\hat{1}|k-1}$

Let $Ey[k]|Y[k-1] = y_{k|\hat{k}-1}$,then

. $Ex[k]|Y[k-1] = x_{k|\hat{k}-1}$

$$\begin{bmatrix} \hat{x}_{k+1|K-1} \\ \hat{y}_{k|k-1} \end{bmatrix} = \begin{bmatrix} A[k]\hat{x}_{k|k-1} + B[k]u[k] \\ C[k]\hat{x}_{k|k-1} \end{bmatrix}$$

- **Consider error state vector**

$\tilde{x}[k] = x[k] - \hat{X}_{k|k-1}$

$\tilde{x}_{k+1} = x[k+1] - x_{k+\hat{1}|k-1}$, and substitute them into (1)yields

$\tilde{y}[k] - y_{k|\hat{k}-1}$

$$\begin{bmatrix} \tilde{x_{k+1}} \\ \tilde{y_k} \end{bmatrix} = \begin{bmatrix} A[k]\tilde{x}_k + v[k] \\ C[k]\tilde{x}_k + \omega[k] \end{bmatrix}$$

- **Compute covariance matrix**

$$\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} = \mathrm{E}\left\{ \left(\begin{bmatrix} x[k+1] \\ y[k] \end{bmatrix} - \begin{bmatrix} \hat{x}_{k+1} \\ \hat{y}_k \end{bmatrix} \right) \left(\begin{bmatrix} x[k+1] \\ y[k] \end{bmatrix} - \begin{bmatrix} \hat{x}_{k+1} \\ \hat{y}_k \end{bmatrix} \right)^T \middle| Y_{k-1} \right\}$$

$$\Sigma = \mathrm{E}\left\{ \begin{bmatrix} \tilde{x}_{k+1} \\ \tilde{y}_k \end{bmatrix} \begin{bmatrix} \tilde{x}_{k+1}^T & \tilde{y}_k^T \end{bmatrix} \middle| Y_{k-1} \right\}$$

$$\Sigma = \mathrm{E}\left\{ \begin{bmatrix} \tilde{x}_{k+1}\tilde{x}_{k+1}^T & \tilde{x}_{k+1}\tilde{y}_k^T \\ \tilde{y}_k\tilde{x}_{k+1}^T & \tilde{y}_k\tilde{y}_k^T \end{bmatrix} \middle| Y_{k-1} \right\}$$

$$\Sigma = \begin{bmatrix} \mathrm{E}\left\{\tilde{x}_{k+1}\tilde{x}_{k+1}^T | Y_{k-1}\right\} & \mathrm{E}\left\{\tilde{x}_{k+1}\tilde{y}_k^T | Y_{k-1}\right\} \\ \mathrm{E}\left\{\tilde{y}_k\tilde{x}_{k+1}^T | Y_{k-1}\right\} & \mathrm{E}\left\{\tilde{y}_k\tilde{y}_k^T | Y_{k-1}\right\} \end{bmatrix}$$

Recall error state vector: $\begin{bmatrix} \hat{x}_{k+1|k-1} \\ \hat{y}_{k|k-1} \end{bmatrix} = \begin{bmatrix} A[k]\hat{x}_{k|k-1} + B[k]u[k] \\ C[k]\hat{x}_{k|k-1} \end{bmatrix}$

$$\begin{bmatrix} \tilde{x}_{k+1} \\ \tilde{y}_k \end{bmatrix} = \begin{bmatrix} A[k]\tilde{x}_k + v[k] \\ C[k]\tilde{x}_k + w[k] \end{bmatrix}$$

$$\begin{aligned} \mathrm{E}\left\{\tilde{x}[k+1]\tilde{x}^T[k+1]\right\} &= \mathrm{E}\left\{ \left(x[k+1] - \hat{x}_{k+1|k-1}\right) \left(x[k+1] - \hat{x}_{k+1|k-1}\right)^T \right\} \\ &= \mathrm{E}\left\{ A[k]\tilde{x}[k]\tilde{x}^T[k]A^T[k] + A[k]\tilde{x}[k]v_k^T + v_k\tilde{x}^T[k]A^T[k] + v_k v_k^T \right\} \\ &= A_k P_{k|k-1} A_k^T + Q_k \end{aligned}$$

$$\Sigma = \begin{bmatrix} A_k P_{k|k-1} A_k^T + Q_k & A_k P_{k|k-1} C_k^T \\ C_k P_{k|k-1} A^T & C_k P_{k|k-1} C_k^T + R_k \end{bmatrix}$$

- **Conditional expectation for Gaussian pdf**

$$\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} = \begin{bmatrix} E\{(x-\bar{x})(x-\bar{x})^T\} & E\{(x-x)(y-\bar{y})^T\} \\ E\{(y-\bar{y})(x-x)^T\} & E\{(y-\bar{y})(y-\bar{y})^T\} \end{bmatrix}$$

$$A_0\Sigma_{yy} = \Sigma_{xy} \qquad A_0 = \Sigma_{xy}\Sigma_{yy}^{-1}$$

$$\hat{x} = E\{x|y\} = \bar{x} + A_0(y-\bar{y})$$

$$E\{[x-E\{x|y\}][x-E\{x|y\}]^T|y\} = \Sigma_{xx} - A_0\Sigma_{yx}$$

Since $\Sigma_{xy}^T = \Sigma_{yy}^T A_0^T$, the variance is given by $\Sigma_{xx} - A_0\Sigma_{yy}A_0^T$

If we consider that $X = x[k+1], Y = y[k]$, then

$z = \hat{x}_{k+1|k}$ can be calculated as

$\bar{x} = A_k\hat{x}_{k|k-1}$

$\bar{y} = C_k\hat{x}_{k|k-1}$

$\Sigma_{yy} = C_kP_{k|k-1}C_k^T + R$

$\Sigma_{xy} = A_kP_{k|k-1}C_k^T$

and

$z = \bar{x} + \Sigma_{xy}\Sigma_{yy}^{-1}(y[k]-\bar{y})$

$= A_k(\hat{x}_{k|k-1} + P_{k|k-1}C_k^T(C_kP_{k|k-1}C_k^T + R)^{-1}(y[k]-\bar{y}))$

$\Sigma_{zz} = A_kP_{k|k-1}A_k^T + Q - A_kP_{k|k-1}C_k^T\left(C_kP_{k|k-1}C_k^T + R\right)^{-1}C_kP_{k|k-1}A^T$

$= A_k(P_{k|k-1} - P_{k|k-1}C_k^T\left(C_kP_{k|k-1}C_k^T + R\right)^{-1}C_kP_{k|k-1})A^T + Q_k$

$= A_k(P_{k|k-1} - W_kC_kP_{k|k-1})A_k^T + Q_k$

If we define a gain as

$$W_k = P_{k|k-1}C_k^T\left(C_kP_{k|k-1}C_k^T + R_k\right)^{-1}$$

$\hat{x}_{k|k}$ and $P_{k|k}$ are defined as

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + W_k(y_k - C_k\hat{x}_{k|k-1})$$

$$P_{k|k} = P_{k|k-1} - W_kC_kP_{k|k-1}$$

The optimal value is obtained as

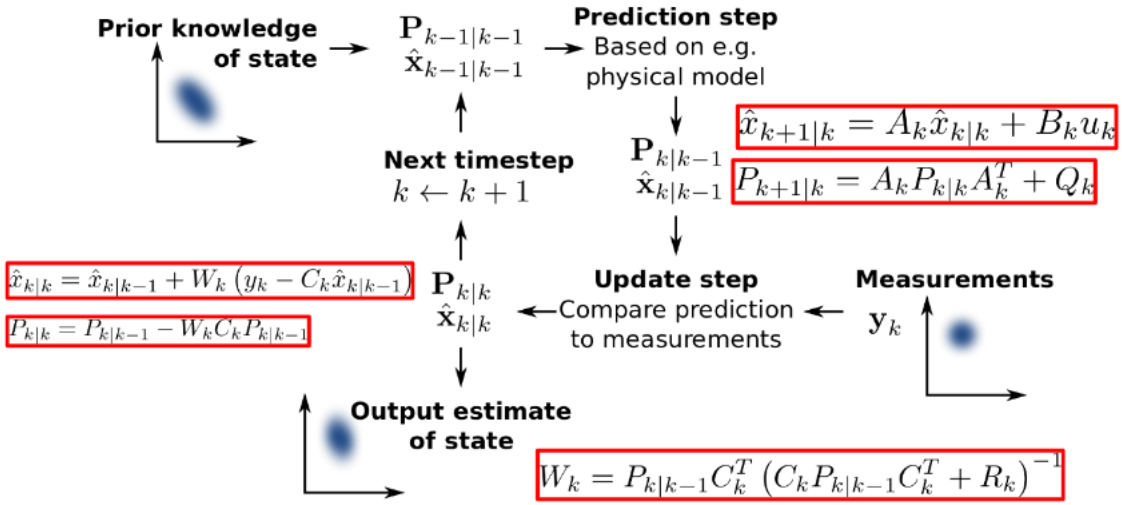
$$\hat{x}_{k+1|k} = A_k\hat{x}_{k|k}$$

$$P_{k+1|k} = A_kP_{k|k}A_k^T + Q_k$$

- **Solution of Quadratic Optimization**
  Min $J = (x-\tilde{x})^TP^{-1}(x-\hat{x}) + (Cx-y)^TR^{-1}(Cx-y)$
  $\hat{x} := \hat{x}_{k|k-1}, y := y_k, P := P_{k|k+1}, R := R_k$
  $\frac{\partial J}{\partial x} = 2P^{-1}(x-\hat{x}) + 2C^TR^{-1}(Cx-y) = 0$

  vspace0.3cm

$(P_{-1} + C^T R^{-1} C)x - P^{-1}\hat{x} - C^T R^{-1} y = 0 \rightarrow x = (P^{-1} + C^T R^{-1} C)^{-1})(P^{-1}\hat{x} + C^T R^{-1} y)$
Using inversion lemma,we have
$(P^{-1} + C^T R^{-1} C)^{-1}) = P - PC^T R{-}1(I + CPC^T R^{-1})^{-1} CP = P - PC^T(R + CPC^T)^{-1} CP$
$x = (P{-}PC^T(R{+}CPC^T)^{-1}CP)(P^{-1}\hat{x}{+}C^T R^{-1} y) = (I{-}PC^T(R{+}CPC^T)^{-1}C)(\hat{x}{+}PC^T R^{-1} y)$
Where $K = PC^T(R + CPC^T)^{-1}$
$= \hat{x} - KC\hat{x} + PC^T(I - (R + CPC^T)^{-1} CPC^T)R^{-1} y$
$= \hat{x} - KC\hat{x} + PC^T(R + CPC^T)^{-1}(R + CPC^T - CPC^T)R^{-1} y$
$= \hat{x} - KC\hat{x} + Ky = \hat{x} + K(y - C\hat{x})$



- **Kalman Filter Algorithm**
  Given: $\hat{x}_{0|-1} and P_{0|-1}$

  Compute Kalman Filter gain

  $$W_k = P_{k|k-1}C_k^T\left(C_k P_{k|k-1} C_k^T + R_k\right)^{-1}$$

  Update (modify) predicted value by observation

  $$\hat{x}_{k|k} = \hat{x}_{k|k-1} + W_k\left(y_k - C_k\hat{x}_{k|k-1}\right)$$

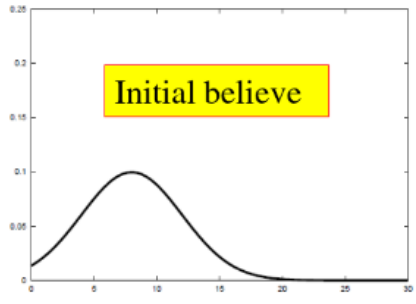
  Compute prediction at the next time step

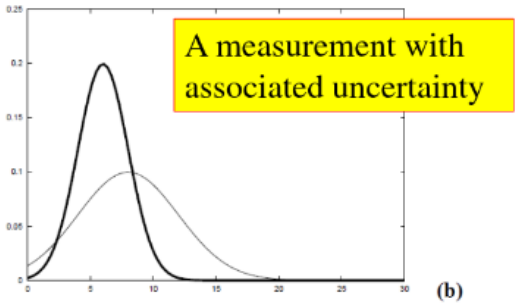
  $$\hat{x}_{k+1|k} = A_k\hat{x}_{k|k} + B_k u_k$$

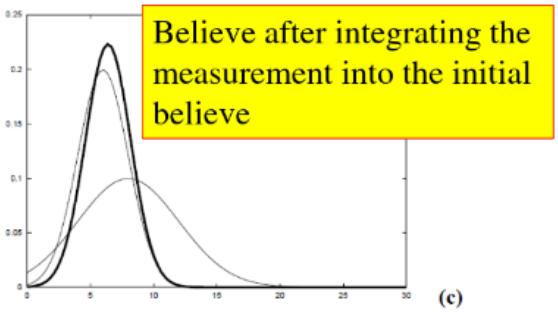
  Update error covariance matrix

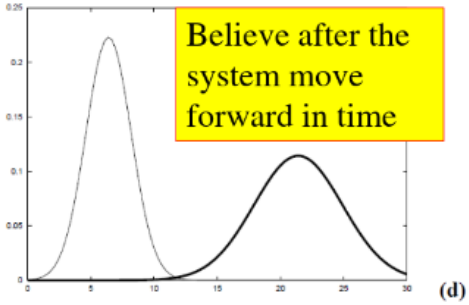
  $$P_{k|k} = P_{k|k-1} - W_k C_k P_{k|k-1}$$

  $$P_{k+1|k} = A_k P_{k|k} A_k^T + Q_k$$









$$W_k = P_{k|k-1} C_k^T \left(C_k P_{k|k-1} C_k^T + R_k \right)^{-1}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + W_k \left(y_k - C_k \hat{x}_{k|k-1} \right)$$

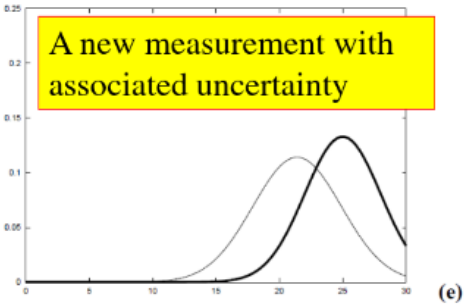
$$P_{k|k} = P_{k|k-1} - W_k C_k P_{k|k-1}$$

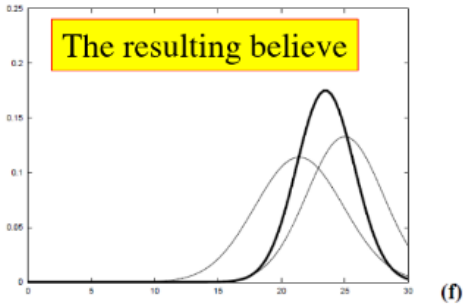
Correction Step

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k} + B_k u_k$$

$$P_{k+1|k} = A_k P_{k|k} A_k^T + Q_k$$

Prediction Step





5.2 Extended Kalman Filter(EKF)

EKFs are common in real-world robotics applications. You'll see them in everything from self-driving cars to drones.

EKFs are useful when:

- You have a robot with sensors attached to it that enable it to perceive the world.
- Your robot's sensors are noisy and aren't 100% accurate (which is always the case).

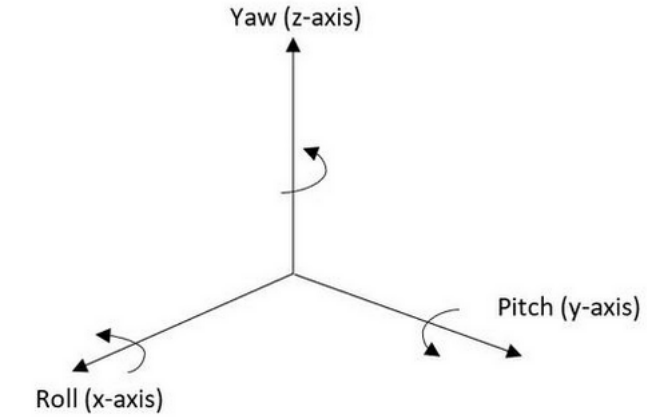
By running all sensor observations through an EKF, you smooth out noisy sensor measurements and can calculate a better estimate of the state of the robot at each timestep t as the robot moves around in the world. By "state", I mean "where is

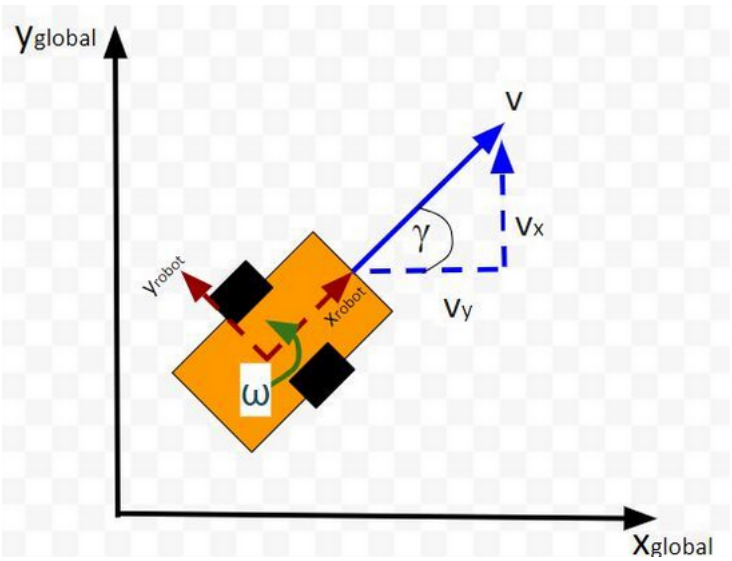
the robot," "what is its orientation," etc.

5.2.1 Prerequisites

In order to understand what an EKF is, you should know what a state space model and an observation model are. These mathematical models are the two main building blocks for EKFs:

- State Space Model
  - **What is a State Space Model?**
    A state space model (often called the state transition model) is a mathematical equation that represents the motion of a robotic system from one timestep to the next. It shows how the current position (e.g. X, Y coordinate) and orientation (yaw (heading) angle γ) of the robot in the world is impacted by changes to the control inputs into the robot (e.g. velocity in meters per second... often represented by the variable v).
    Note: In a real-world scenario, we would need to represent the world the robot is in (e.g. a room in your house, etc.) as an x,y,z coordinate grid.





Using trigonometry: We know that:
$cos(\gamma) = \frac{V_x}{V}$
$\Rightarrow$ linear velocity in the x direction: $V_x = V cos(\gamma)$
$sin(\gamma) = \frac{V_y}{V}$
$\Rightarrow$ linear velocity in the y direction: $V_y = V sin(\gamma)$
angular velocity around the z axis= ω
distance = velocity * time.

  - **Converting the Equations to a State Space Model**

$$\begin{bmatrix} x_t \\ y_t \\ \gamma_t \end{bmatrix} = A_{t-1} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \gamma_{t-1} \end{bmatrix} + B_{t-1} \begin{bmatrix} V_{t-1} \\ \omega_{t-1} \end{bmatrix}$$

- **How to Calculate the A Matrix**

$$\begin{bmatrix} x_t \\ y_t \\ \gamma_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + V_{t-1}cos(\omega_{t-1}) * dt \\ y_{t-1} + V_{t-1}sin(\omega_{t-1}) * dt \\ \gamma_{t-1} + \omega_{t-1} * dt \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

$$A_{t-1} = \begin{bmatrix} \frac{\partial f_1}{\partial x_{t-1}} & \frac{\partial f_1}{\partial y_{t-1}} & \frac{\partial f_1}{\partial \gamma_{t-1}} \\ \frac{\partial f_2}{\partial x_{t-1}} & \frac{\partial f_2}{\partial y_{t-1}} & \frac{\partial f_2}{\partial \gamma_{t-1}} \\ \frac{\partial f_3}{\partial x_{t-1}} & \frac{\partial f_3}{\partial y_{t-1}} & \frac{\partial f_3}{\partial \gamma_{t-1}} \end{bmatrix}$$

The Result when you calculate any partial derivative

$$\Rightarrow A_{t-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **How to Calculate the B Matrix**

$$\begin{bmatrix} \frac{\partial f_1}{\partial V_{t-1}} & \frac{\partial f_1}{\partial \omega_{t-1}} \\ \frac{\partial f_2}{\partial V_{t-1}} & \frac{\partial f_2}{\partial \omega_{t-1}} \\ \frac{\partial f_3}{\partial V_{t-1}} & \frac{\partial f_3}{\partial \omega_{t-1}} \end{bmatrix}$$

The result When we calculate any partial derivative

$$B_{t-1} = \begin{bmatrix} cos(\gamma_{t-1}) * dt & 0 \\ sin(\gamma_{t-1}) * dt & 0 \\ 0 & dt \end{bmatrix}$$

- **Adding Process Noise**

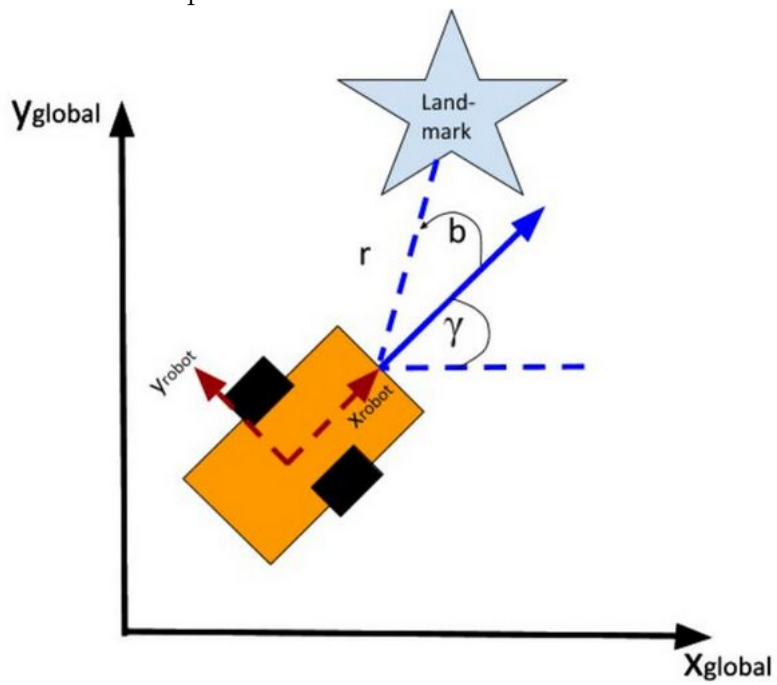
$$\begin{bmatrix} x_t \\ y_t \\ \gamma_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \gamma_{t-1} \end{bmatrix} + \begin{bmatrix} cos(\gamma_{t-1}) * dt & 0 \\ sin(\gamma_{t-1}) * dt & 0 \\ 0 & dt \end{bmatrix} \begin{bmatrix} V_{t-1} \\ \omega_{t-1} \end{bmatrix} + \begin{bmatrix} noise_{t-1} \\ noise_{t-1} \\ noise_{t-1} \end{bmatrix}$$

- Observation Model
  - **What is an Observation Model?** An observation model (also known as measurement model or sensor model) is a mathematical equation that represents a vector of predicted sensor measurements y at time t as a function of the state of a robotic system x at time t, plus some sensor noise (because sensors aren't 100% perfect) at time t, denoted by vector ω_t.
    Here is the formal equation: $y_t = Hx_t + W_t$
    Because a mobile robot in 3D space has three states [x, y, yaw angle], in vector format, the observation model above becomes:

$$\begin{bmatrix} y_1^t \\ y_1^t \\ \vdots \\ y_n^t \end{bmatrix} = H_{nx3}^t \begin{bmatrix} x_t \\ y_t \\ \gamma_t \end{bmatrix} + \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{bmatrix}$$

where:
_t = current time
_y vector (at left) = n predicted sensor observations at time t.
_n = number of sensor observations (i.e. measurements) at time t
_H = measurement matrix (used to convert the state at time t into predicted sensor observations at time t) that has n rows and 3 columns (i.e. a column for each state variable).
_w = the noise of each of the n sensor observations (you often get this sensor error information from the datasheet that comes when you purchase a sensor)

– **How to Calculate the H Matrix**
The measurement matrix H is used to convert the predicted state estimate at time t into predicted sensor measurements at time t.



Using the Pythangorean theorem and trigonometry:we get the following equations for the range r and bearing b:
$r = \sqrt{(x_t - x_{landmark})^2 + (y_t - y_{landmark})^2}$
$b = atan2(y_t - y_{landmark}, x_t - x_{landmark}) - \gamma_t$
Let's put this in matrix form.

$$\begin{bmatrix} r \\ b \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \sqrt{(x_t - x_{landmark})^2 + (y_t - y_{landmark})^2} \\ atan2(y_t - y_{landmark}, x_t - x_{landmark}) - \gamma_t \end{bmatrix}$$

The formula for Ht is as follows:

$$\begin{bmatrix} \frac{\partial r}{\partial x_t} & \frac{\partial r}{\partial y_t} & \frac{\partial r}{\partial \gamma_t} \\ \frac{\partial b}{\partial x_t} & \frac{\partial b}{\partial y_t} & \frac{\partial b}{\partial \gamma_t} \end{bmatrix}$$

So we need to calculate 6 different partial derivatives. Here is what you should get:

$$H_t \begin{bmatrix} \frac{x_t - x_L}{\sqrt{(x_t-x_L)^2+(y_t-y_L)^2}} & \frac{y_t - y_L}{\sqrt{(x_t-x_L)^2+(y_t-y_L)^2}} & 0 \\ \frac{-(y_t-y_L)}{(x_t-x_L)^2+(y_t-y_L)^2} & x_t - x_L(x_t - x_L)^2 + (y_t - y_L)^2 & -1 \end{bmatrix}$$

- **Putting It All Together**
  The final observation model is as follows:

$$\begin{bmatrix} y_1^t \\ y_2^t \end{bmatrix} = \begin{bmatrix} r_t \\ b_t \end{bmatrix} = \begin{bmatrix} \frac{x_t - x_L}{\sqrt{(x_t-x_L)^2+(y_t-y_L)^2}} & \frac{y_t - y_L}{\sqrt{(x_t-x_L)^2+(y_t-y_L)^2}} & 0 \\ \frac{-(y_t-y_L)}{(x_t-x_L)^2+(y_t-y_L)^2} & x_t - x_L(x_t - x_L)^2 + (y_t - y_L)^2 & -1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ \gamma_t \end{bmatrix} + \begin{bmatrix} w_r \\ w_b \end{bmatrix}$$

5.2.2 What is the Extended Kalman Filter?

The Extended Kalman Filter is an algorithm that leverages our knowledge of the physics of motion of the system (i.e. the state space model) to make small adjustments to (i.e. to filter) the actual sensor measurements (i.e. what the robot's sensors actually observed) to reduce the amount of noise, and as a result, generate a better estimate of the state of a system.

- The **state estimate** for the previous timestep t-1
- The **time interval** dt from one timestep to the next
- The **linear and angular velocity** of the car at the previous time step t-1 (i.e. previous control inputs...i.e. commands that were sent to the robot to make the wheels rotate accordingly)
- An **estimate of random noise** (typically small values...when we send velocity commands to the car, it doesn't move exactly at those velocities so we add in some random noise)

It calculates a weighted average of actual sensor measurements at the current timestep t and predicted sensor measurements at the current timestep t to generate a better current state estimate.

5.2.3 What is the Difference Between the Kalman Filter and the Extended Kalman Filter?

- The Extended Kalman Filter was developed to enable the Kalman Filter to be applied to systems that have nonlinear dynamics like our mobile robot.

5.2.4 Extended Kalman Filter Assumptions

EKFs assume you have already derived two key mathematical equations (i.e. models):

- State Space Model:A state space model (often called the state transition model) is a mathematical equation that helps you estimate the state of a system at time t given the state of a system at time t-1.
- Observation Model:An observation model is a mathematical equation that expresses sensor measurements at time t as a function of the state of a robot at time t.

5.2.5 EKF Algorithm

- **Overview**
  On a high-level, the EKF algorithm has two stages, a predict phase and an update (correction phase). Don't worry if all this sounds confusing.
- **Predict Step**
  - Using the state space model of the robotics system, predict the state estimate at time t based on the state estimate at time t-1 and the control input applied at time t-1.

  - Predict the state covariance estimate based on the previous covariance and some noise.
- **Update (Correct) Step**
  - Calculate the difference between the actual sensor measurements at time t minus what the measurement model predicted the sensor measurements would be for the current timestep t.
  - Calculate the measurement residual covariance.
  - Calculate the near-optimal Kalman gain.
  - Calculate an updated state estimate for time t.
  - Update the state covariance estimate for time t.

5.2.6 EKF Algorithm Step-by-Step

- **Step1:Initialization**
  - For the first iteration of EKF, we start at time k. In other words, for the first run of EKF, we assume the current time is k.
  - We initialize the state vector and control vector for the previous time step k-1.
- **Step2:Predicted State Estimate**
  - **Predicted state estimate:** $\hat{\mathbf{X}}_{\mathbf{k|k-1}} = \mathbf{f}(\hat{\mathbf{X}}_{\mathbf{k-1|k-1}}, \mathbf{u_k})$
  - We use the state space model, the state estimate for timestep k-1, and the control input vector at the previous time step (e.g. at time k-1) to predict what the state would be for time k (which is the current timestep).
  - That equation above is the same thing as our equation below. Remember that we used t in my earlier tutorials.
    $\mathbf{X_t} = \hat{\mathbf{X}}_{\mathbf{k|k-1}} = \mathbf{A_{k-1}x_{k-1}} + \mathbf{B_{k-1}U_{k-1}} + \mathbf{V_{k-1}}$
- **Step3:Predicted Covariance of the State Estimate**
  - **Predicted covariance estimate:** $\mathbf{P_{k|k-1}} = \mathbf{F_k P_{k-1|k-1} F_k^T} + \mathbf{Q_k}$
  - $\mathbf{P_{k-1|k-1}}$ is a square matrix. It has the same number of rows (and columns) as the number of states in the state vector x.
  - P (or, commonly, sigma $\sum$) is a 3×3 matrix. The P matrix has variances on the diagonal and covariances on the off-diagonal.
  - it is a matrix that represents an estimate of the accuracy of the state estimate we made in Step 2.
  - we initialize $\mathbf{P_{k-1|k-1}}$ to some guessed values (e.g. 0.1 along the diagonal part of the matrix and 0s elsewhere).

$$\mathbf{P_{k-1|k-1}} = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$$

  - $\mathbf{F_k}$**and**$\mathbf{F_k^T}$:are equivalent to A_{t-1} and A_{t-1}^T, respectively, from my state space model tutorial.

$$\mathbf{F_k} = \mathbf{F_k^T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - $\mathbf{Q_k}$:is the state model noise covariance matrix. It is also a 3×3 matrix in our running robot car example because there are three states.

$$\mathbf{Q_k} = \begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,\gamma) \\ Cov(y,x) & Cov(y,y) & Cov(y,\gamma) \\ Cov(\gamma,x) & Cov(\gamma,y) & Cov(\gamma,\gamma) \end{bmatrix}$$

- The covariance between two variables that are the same is actually the variance. For example, $\mathbf{Cov(x, x) = Var(x)}$.
- Variance measures the deviation from the mean for points in a single dimension (i.e. x values, y values, or yaw angle values).
- We can start by letting Q be the identity matrix and tweak the values through trial and error.

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

- when Q is large, it means we trust our actual sensor observations more than we trust our predicted sensor measurements from the observation model. . . more on this later in this tutorial.

- **Step4:Innovation or Measurement Residual**
  - **Innovation or measurement residual** $\tilde{y_k} = z_k - h(\hat{x}_{k|k-1})$
  - we calculate the difference between actual sensor observations and predicted sensor observations.
  - $\mathbf{z_k}$ is the observation vector. It is a vector of the actual readings from our sensors at time k.Matrix in z_k in mobile robot:

$$z_k = \begin{bmatrix} x_k \\ y_k \\ \gamma_k \end{bmatrix}$$

  - $h(\hat{x}_{k|k-1})$ is our observation model. It represents the predicted sensor measurements at time k given the predicted state estimate at time k from Step 2. That hat symbol above x means "predicted" or "estimated".

$$h(\hat{x}_{k|k-1}) = H_k\hat{x}_{k|k-1} + w_k$$

  - We use the:
    measurement matrix $\mathbf{H_k}$ (which is used to convert the predicted state estimate at time k into predicted sensor measurements at time k),

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

    predicted state estimate $\mathbf{\hat{X}_{k|k-1}}$ that we calculated in Step 2,

$$\mathbf{\hat{X}_{k|k-1}} = AnswerFromStep2.$$

    and a sensor noise assumption $\mathbf{w_k}$ (which is a vector with the same number of elements as there are sensor measurements)

- **Step5:Innovation (or residual) Covariance**
  - Innovatiion (or residual) covariance : $\mathbf{S_k = H_k P_{k|k-1} H_k^T + R_k}$
  - use the predicted covariance of the state estimate $\mathbf{P_{k|k-1}}$ from Step 3.
  - The measurement matrix **Hk** and its transpose.
  - **Rk** (sensor measurement noise covariance matrix. . . which is a covariance matrix that has the same number of rows as sensor measurements and same number of columns as sensor measurements)

- To start out by making $\mathbf{R_k}$ the identity matrix. You can then tweak it through trial and error

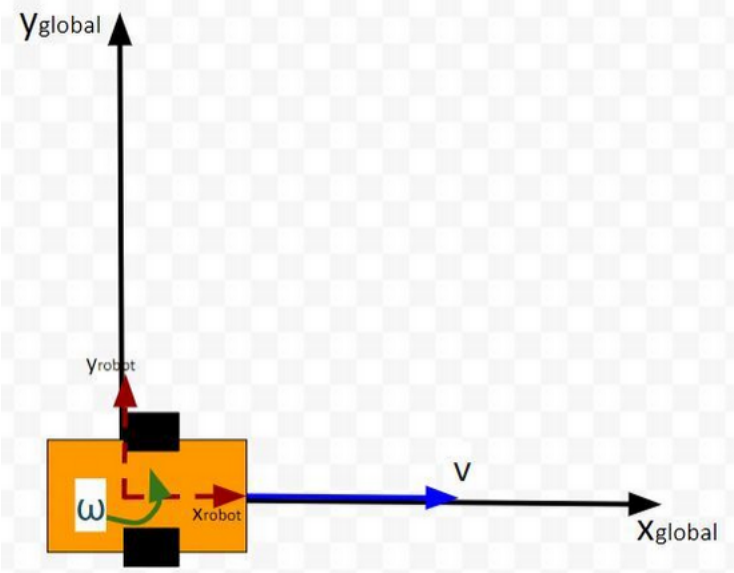
$$\mathbf{R_k} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Step6:Near-optimal Kalman Gain**
  - Near-optimal kalman Gain: $\mathbf{K_k = P_{k|k-1} H_k^T S_k^{-1}}$
  - If sensor measurement noise is large, then K approaches 0, and sensor measurements will be mostly ignored.
  - If prediction noise (using the dynamical model/physics of the system) is large, then K approaches 1, and sensor measurements will dominate the estimate of the state [x,y,yaw angle].
- **Step7:Updated State Estimate**
  - Update state estimate: $\mathbf{\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k}$
    In this step, we calculate an updated (corrected) state estimate based on the values from:
    _Step 2 (predicted state estimate for current time step k),
    _Step 6 (near-optimal Kalman gain from 6),
    _Step 4 (measurement residual).
- **Step8:Updated Covariance of the State Estimate**

$$\mathbf{P_{k|k} = (I - K_k H_k) P_{k|k-1}}$$

5.2.7 Python Code for the Extended Kalman Filter

Let's assume our robot starts out at the origin (x=0, y=0), and the yaw angle is 0 radians.



We then apply a forward linear velocity v of 4.5 meters per second at time k-1 and an angular velocity ω of 0 radians per second.
Here is the code:

```python
import numpy as np

# Author: Addison Sears-Collins
# https://automaticaddison.com
# Description: Extended Kalman Filter example (two-wheeled mobile robot)

# Supress scientific notation when printing NumPy arrays
np.set_printoptions(precision=3,suppress=True)

# A matrix
# 3x3 matrix -> number of states x number of states matrix
# Expresses how the state of the system [x,y,yaw] changes
# from k-1 to k when no control command is executed.
# Typically a robot on wheels only drives when the wheels are told to turn.
# For this case, A is the identity matrix.
# A is sometimes F in the literature.
A_k_minus_1 = np.array([[1.0,  0,   0],
                        [  0,1.0,   0],
                        [  0,  0, 1.0]])

# Noise applied to the forward kinematics (calculation
# of the estimated state at time k from the state
# transition model of the mobile robot). This is a vector
# with the number of elements equal to the number of states
process_noise_v_k_minus_1 = np.array([0.01,0.01,0.003])

# State model noise covariance matrix Q_k
# When Q is large, the Kalman Filter tracks large changes in
# the sensor measurements more closely than for smaller Q.
# Q is a square matrix that has the same number of rows as states.
Q_k = np.array([[1.0,   0,   0],
                [  0, 1.0,   0],
                [  0,   0, 1.0]])

# Measurement matrix H_k
# Used to convert the predicted state estimate at time k
# into predicted sensor measurements at time k.
# In this case, H will be the identity matrix since the
# estimated state maps directly to state measurements from the
# odometry data [x, y, yaw]
# H has the same number of rows as sensor measurements
# and same number of columns as states.
H_k = np.array([[1.0,  0,   0],
                [  0,1.0,   0],
                [  0,  0, 1.0]])

# Sensor measurement noise covariance matrix R_k
# Has the same number of rows and columns as sensor measurements.
# If we are sure about the measurements, R will be near zero.
R_k = np.array([[1.0,   0,    0],
                [  0, 1.0,    0],
                [  0,    0, 1.0]])

# Sensor noise. This is a vector with the
# number of elements equal to the number of sensor measurements.
sensor_noise_w_k = np.array([0.07,0.07,0.04])

def getB(yaw, deltak):
    """
    Calculates and returns the B matrix
    3x2 matix -> number of states x number of control inputs
    The control inputs are the forward speed and the
    rotation rate around the z axis from the x-axis in the
    counterclockwise direction.
    [v,yaw_rate]
    Expresses how the state of the system [x,y,yaw] changes
    from k-1 to k due to the control commands (i.e. control input).
    :param yaw: The yaw angle (rotation angle around the z axis) in rad
    :param deltak: The change in time from time step k-1 to k in sec
    """
    B = np.array([  [np.cos(yaw)*deltak, 0],
                                [np.sin(yaw)*deltak, 0],
                                [0, deltak]])
    return B

def ekf(z_k_observation_vector, state_estimate_k_minus_1,
        control_vector_k_minus_1, P_k_minus_1, dk):
    """
    Extended Kalman Filter. Fuses noisy sensor measurement to
    create an optimal estimate of the state of the robotic system.

    INPUT
        :param z_k_observation_vector The observation from the Odometry
            3x1 NumPy Array [x,y,yaw] in the global reference frame
            in [meters,meters,radians].
        :param state_estimate_k_minus_1 The state estimate at time k-1
            3x1 NumPy Array [x,y,yaw] in the global reference frame
            in [meters,meters,radians].
        :param control_vector_k_minus_1 The control vector applied at time k-1
            3x1 NumPy Array [v,v,yaw rate] in the global reference frame
            in [meters per second,meters per second,radians per second].
        :param P_k_minus_1 The state covariance matrix estimate at time k-1
            3x3 NumPy Array
        :param dk Time interval in seconds

    OUTPUT
        :return state_estimate_k near-optimal state estimate at time k
            3x1 NumPy Array ---> [meters,meters,radians]
        :return P_k state covariance_estimate for time k
            3x3 NumPy Array
    """

    ######################### Predict #############################
    # Predict the state estimate at time k based on the state
    # estimate at time k-1 and the control input applied at time k-1.
    state_estimate_k = A_k_minus_1 @ (
            state_estimate_k_minus_1) + (
            getB(state_estimate_k_minus_1[2],dk)) @ (
            control_vector_k_minus_1) + (
            process_noise_v_k_minus_1)

    print(f'State Estimate Before EKF={state_estimate_k}')

    # Predict the state covariance estimate based on the previous
    # covariance and some noise
    P_k = A_k_minus_1 @ P_k_minus_1 @ A_k_minus_1.T + (
            Q_k)

    ################### Update (Correct) ##########################
    # Calculate the difference between the actual sensor measurements
    # at time k minus what the measurement model predicted
    # the sensor measurements would be for the current timestep k.
    measurement_residual_y_k = z_k_observation_vector - (
            (H_k @ state_estimate_k) + (
            sensor_noise_w_k))

    print(f'Observation={z_k_observation_vector}')

    # Calculate the measurement residual covariance
    S_k = H_k @ P_k @ H_k.T + R_k

    # Calculate the near-optimal Kalman gain
    # We use pseudoinverse since some of the matrices might be
    # non-square or singular.
    K_k = P_k @ H_k.T @ np.linalg.pinv(S_k)

    # Calculate an updated state estimate for time k
    state_estimate_k = state_estimate_k + (K_k @ measurement_residual_y_k)

    # Update the state covariance estimate for time k
    P_k = P_k - (K_k @ H_k @ P_k)

    # Print the best (near-optimal) estimate of the current state of the robot
    print(f'State Estimate After EKF={state_estimate_k}')

    # Return the updated state and covariance estimates
    return state_estimate_k, P_k

def main():

    # We start at time k=1
    k = 1

    # Time interval in seconds
    dk = 1

    # Create a list of sensor observations at successive timesteps
    # Each list within z_k is an observation vector.
    z_k = np.array([[4.721,0.143,0.006], # k=1
                    [9.353,0.284,0.007], # k=2
                    [14.773,0.422,0.009],# k=3
                    [18.246,0.555,0.011], # k=4
                    [22.609,0.715,0.012]])# k=5

    # The estimated state vector at time k-1 in the global reference frame.
    # [x_k_minus_1, y_k_minus_1, yaw_k_minus_1]
    # [meters, meters, radians]
    state_estimate_k_minus_1 = np.array([0.0,0.0,0.0])

    # The control input vector at time k-1 in the global reference frame.
    # [v, yaw_rate]
    # [meters/second, radians/second]
    # In the literature, this is commonly u.
    # Because there is no angular velocity and the robot begins at the
    # origin with a 0 radians yaw angle, this robot is traveling along
    # the positive x-axis in the global reference frame.
    control_vector_k_minus_1 = np.array([4.5,0.0])

    # State covariance matrix P_k_minus_1
    # This matrix has the same number of rows (and columns) as the
    # number of states (i.e. 3x3 matrix). P is sometimes referred
    # to as Sigma in the literature. It represents an estimate of
    # the accuracy of the state estimate at time k made using the
    # state transition matrix. We start off with guessed values.
    P_k_minus_1 = np.array([[0.1,  0,   0],
                                                [  0,0.1,   0],
                                                [  0,  0, 0.1]])

    # Start at k=1 and go through each of the 5 sensor observations,
    # one at a time.
    # We stop right after timestep k=5 (i.e. the last sensor observation)
    for k, obs_vector_z_k in enumerate(z_k,start=1):

        # Print the current timestep
        print(f'Timestep k={k}')
```

```
195
196         # Run the Extended Kalman Filter and store the
197         # near-optimal state and covariance estimates
198         optimal_state_estimate_k, covariance_estimate_k = ekf(
199             obs_vector_z_k, # Most recent sensor measurement
200             state_estimate_k_minus_1, # Our most recent estimate of the state
201             control_vector_k_minus_1, # Our most recent control input
202             P_k_minus_1, # Our most recent state covariance matrix
203             dk) # Time interval
204
205         # Get ready for the next timestep by updating the variable values
206         state_estimate_k_minus_1 = optimal_state_estimate_k
207         P_k_minus_1 = covariance_estimate_k
208
209         # Print a blank line
210         print()
211
212 # Program starts running here with the main method
```

6. **MPC(Model Predictive Control)**

5. 1 **Definition**

Use a dynamical model of the process to predict its future evolution and choose the "best" control action.

- MPC problem= find the best control sequence over a future horizon of N steps.

$$min_{u_0,...,u_{n-1}} \sum_{k=0}^{N-1} ||Y_k - r(t)||_2^2 + \rho||U_k - U_r(t)||_2^2$$

Such that :
prediction model

$$X_{k+1} = f(X_k, U_k)$$
$$Y_k = g(X_k)$$

constraints

$$U_{min} <= U_k <= U_{max}$$
$$Y_{min} <= Y_k <= Y_{max}$$

State feedbadk

$$X_0 = X(t)$$

- Numerical optimization problem
  i. Estimate current state X(t).
  ii. Optimize wrt$u_0, ..., u_{N-1}$
  iii. Only apply optimal U_0 as input U(t) Repeat at all time steps t

5. 2 LINEAR_MPC

- Linear prediction model: $\begin{Bmatrix} X_{k+1} = AX_k + BU_k \\ Y_k = CX_k \end{Bmatrix}$

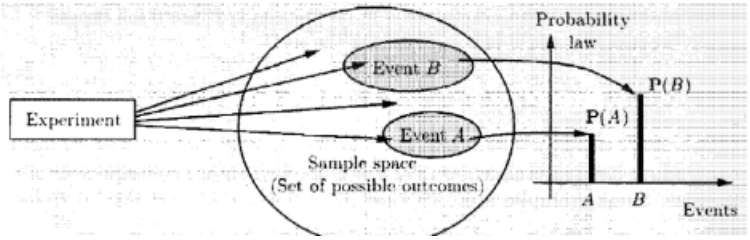
III ESTIMATE

1. **Statistic**

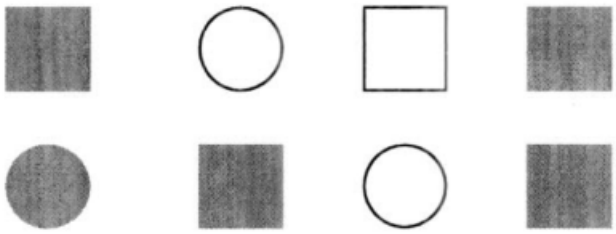
1.1 **BasicProbability**

Probability: measure a uncertainly.
Event: a collection of possible outcomes.

Sample space: set of all possible outcomes of an experiment



Consider 8 shapes below:



We have 3 circles $P(\text{circle}) = \frac{3}{8}$

What is the probability that one of the shapes is a gray circle? $P(\text{gray, circle}) = \frac{1}{8}$

What is the probability that of the three circles, only one is gray? $P(\text{gray}|\text{circle}) = \frac{1}{3}$

The conditional probability of A given B is defined as

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

Priori probability	$P(\text{A}), P(\text{B})$	Known in advance
Posteriori probability	$P(\text{A}\|\text{B})$	a probability given the fact that some information about a possibly related event is already known.
Joint probability	$P(\text{A}, \text{B})$	

We can use this formula to calculate $P(\text{gray}|\text{circle})$

$$P(\text{gray}|\text{circle}) = \frac{P(\text{gray}, \text{circle})}{P(\text{circle})} = \frac{\frac{1}{8}}{\frac{3}{8}} = \frac{1}{3}$$

1.1. 1 Probability

_complement: $\bar{A} or A^c = \{\omega \epsilon S : \omega not \epsilon A\}$
_Union: $A \cup B = \{\omega \epsilon S : \omega \epsilon A or \omega \epsilon B\}$
_Intersection: $A \cap B = \{\omega \epsilon A and \omega \epsilon B\}$
+Probability set function:

- $P(A) = 1 - P(\bar{A})$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A_1 \cup A_2 ... \cup A_n) = p_1 - p_2 + ... + (-1)^{n-1} p_n$
  where $p_1 = P(A_1) + P(A_2) + ...; p_2 = P(A_1 \cap A_2) + P(A_2 \cap A_3) + ...; p_3 = P(A_1 \cap A_2 \cap A_3) + ...$
- Opermutation: $P_n^r = \frac{n!}{(n-r)!}$
- Combination: $C_n^r = \frac{n!}{n!(n-r)!}$

- Conditional probability of A given that B has occured: $P(A/B) = \frac{P(A \cap B}{P(B)}$
- The law of total probability: $P(B) = \sum_{i=1}^{k} P(B/A_i)P(A_i) = P(B/A_i)P(A_1) + ... + P(B/A_k)P(A_k)$
- Bayes's theorem: $P(A_i/B) = \frac{P(A_i \cap B)}{P(B)} = \frac{P(B/A_i)P(Ai)}{\sum_{i=1}^{k} P(B/A_i)P(A_i)}$
- Multually independent: $P(A_{i1} \cap A_{i2} \cap ... \cap A_{ik}) = P(A_{i1})P(A_{i2})$

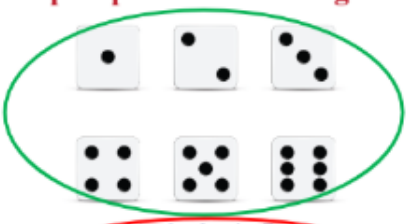
1.1. 2 Random variable

Random Variable(RV): a function that maps a random event to a numerical values.RV will always be random and will never be equal to a specific value.

Example 1

Fig. 1 Rolling a die

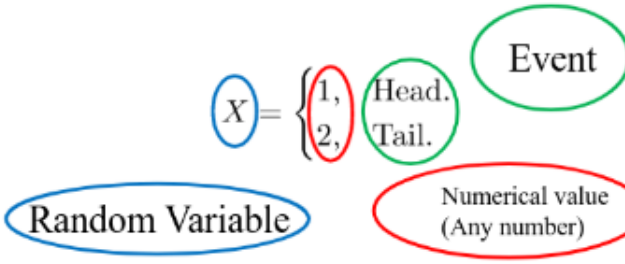
Sample space while rolling a die



$= \{1, 2, 3, 4, 5, 6\}$

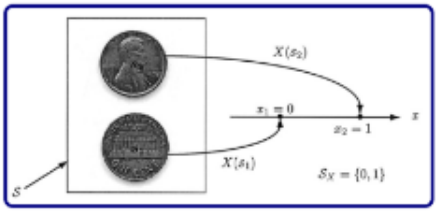
Example 2

Fig. 2 Tossing a coin



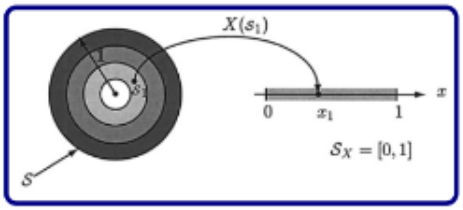
Discrete RV

Finite number
Probability Mass Function
Cumulative Distribution Function
Expectation
Variance



Continuous RV

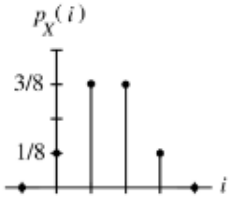
Infinite number
Probability Density Function
Cumulative Distribution Function
Expectation
Variance



Probability Mass Function (Discrete RV)

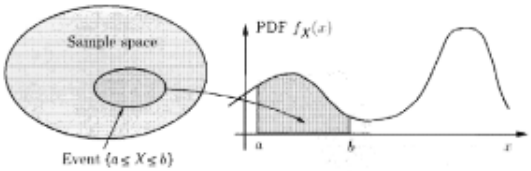
The probabilities of the values that it can take.

$$X(\omega) := \begin{cases} 0, & \omega = \text{TTT}, \\ 1, & \omega \in \{\text{TTH}, \text{THT}, \text{HTT}\}, \\ 2, & \omega \in \{\text{THH}, \text{HTH}, \text{HHT}\}, \\ 3, & \omega = \text{HHH}. \end{cases}$$



$$p_X(x_i) := P(X = x_i)$$
$$0 \le p_X(x_i) \le 1$$
$$\sum_i p_X(x_i) = 1$$

Probability Density Function (Continuous RV)

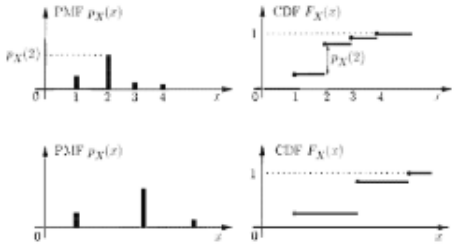


$$P(a \le X \le b) = \int_a^b f(t)dt$$
$$p_X(x) \ge 0, \quad -\infty < x < \infty$$
$$\int_{-\infty}^{\infty} p_X(x) = 1$$

10

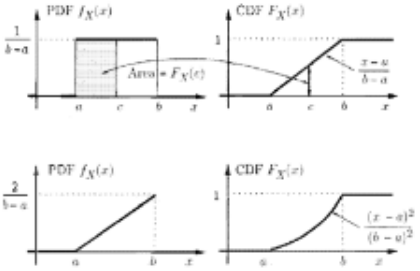
Cumulative Distribution Function (Discrete RV)

It would be desirable to describe all kinds of random variables with a single mathematical concept. This is accomplished with the cumulative distribution function, or CDF for short.



$$F(x) = P(X \le x) = \sum_{i:x_i \le x} p(x_i)$$

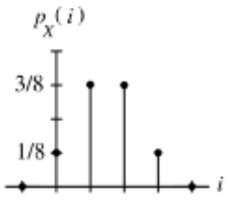
Cumulative Distribution Function (Continuous RV)



$$F(x) = P(X \le x) = \int_{-\infty}^{x} f(t)dt$$

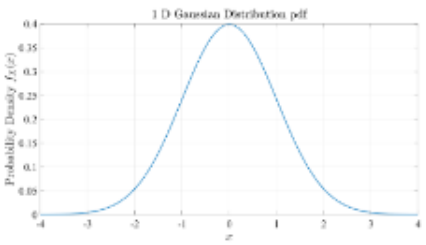
Expectation (Discrete RV)

Expectation can be viewed as the mean value which is the weighted average of the possible values of X, or as the center of gravity of the PMF.



$$\mathrm{E}[X] := \sum_i x_i P(X = x_i)$$

Expectation (Continuous RV)



$$\mathrm{E}[x] := \int_{-\infty}^{-\infty} x f(x)dx$$

Variance (Discrete RV)

Variance (Continuous RV)

The variance tells us how spread out the RV is.

$$\mathrm{Var}[X] = \mathrm{E}\left(X - \overline{X}\right)^2 = \mathrm{E}[X^2] - ([\mathrm{X}])^2$$

The standard deviation is the square root of the variance

$$\sigma_X = \sqrt{\mathrm{var}(X)}$$

Since the standard deviation has the same units as the RV X, it is easier to interpret.

Properties of Mean and Variance

Functions of Random Variables

$$Y = g(X) = \begin{cases} aX + b, \text{for example is linear.} \\ \log X, \text{for example is nonlinear.} \end{cases}$$

In the case of linear, the mean and variance are defined as follows

$$\mathrm{E}[Y] = a\,\mathrm{E}[X] + b$$

$$\mathrm{var}[Y] = a^2\,\mathrm{var}[X]$$

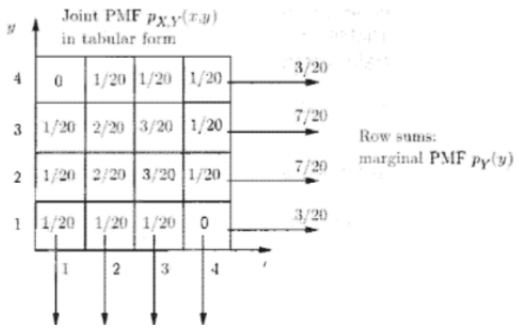
$$\mathrm{E}[g(X)] = \textstyle\sum_x g(X) p_X(x)$$

Joint Probability Mass Function

$$P((X,Y) \in A) := \sum_{(x,y)\in A} p_{X,Y}(x,y)$$

Marginal Probability Mass Function

$$p_X(x) := \sum_y p_{X,Y}(x,y), \quad p_Y(y) := \sum_x p_{X,Y}(x,y)$$

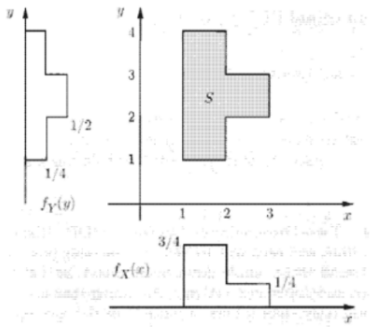


Joint Probability Density Function

$$P((X,Y) \in B) = \iint_{(x,y)\in B} f_{X,Y(x,y)} dxdy$$

Marginal Probability Density Function

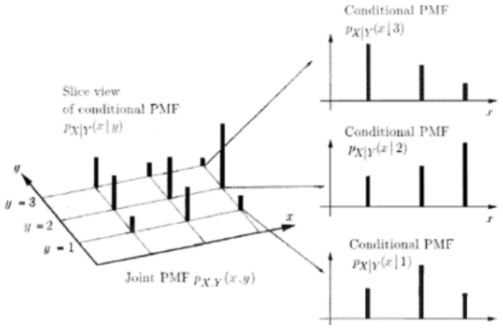
$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dy, \quad f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dx$$

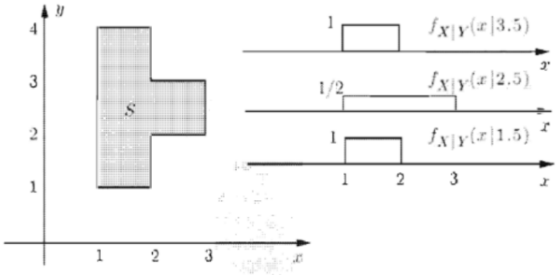


Joint PDF and the resulting Marginal PDF

Visualization of the conditional PMF

Visualization of the conditional PDF





Independence of Discrete Random Variables

Let X and Y be jointly discrete random variable.

- X and Y are independent if for all paris (x,y), the events {X=x} and Y=y are independent, or equivalently

$$p_{X,Y}(x,y) = p_X(x)p_Y(y), forallx, y.$$

- If X and Y are independent,then E[XY]=E[X]E[Y]
- For any function g and h, the random variable g(X) and h(Y) are independent,and we have

$$E[g(X)h(Y)] = E[g(X)]E[h(Y)]$$

- if X and Y are independent,then

$$var(X+Y) = var(X) + var(Y)$$

Independence of Continuous Random Variables

Let X and Y b jointly continuous random variable.

- X and Y are independent if for all paris (x,y), the events {X=x} and Y=y are independent, or equivalently

$$f_{X,Y}(x,y) = f_X(x)f_Y(y), forallx, y.$$

- If X and Y are independent,then E[XY]=E[X]E[Y]
- For any function g and h, the random variable g(X) and h(Y) are independent,and we have

$$E[g(X)h(Y)] = E[g(X)]E[h(Y)]$$

- if X and Y are independent,then

$$var(X+Y) = var(X) + var(Y)$$

Correlation and Covariance

- The correlation is important because it determines when two random variables are linealy related;namely,when one is linear function of the other.
- A pair of random variables is said to be uncorrelated if their correlation coefficient is zero.

$$CorrelationE[XY]$$

- The correlation coefficient of random variables X and Y is defined to be the correlation of their normalized versions

$$\rho_{XY} : E\left[\left(\frac{X-\bar{x}}{\sigma_x}\right)\left(\frac{Y-\bar{y}}{\sigma_y}\right)\right]$$

$$If\rho_{XY}is0, XandYaresaidtobeuncorrelated.$$

- Covariance between two random varibles X and Y is defined as

$$cov(X,Y) := E[(X-\bar{x})(Y-\bar{y})]$$

- We can rewrite the correlation coefficient as

$$\rho_{XY} = \frac{cov(X,Y)}{\sqrt{var(x)var(y)}}, -1 \leq \rho_{XY} \leq 1$$

Hence,X and Y are uncorrelated if and only if their covariance is zero.If X and Y are indpendent,they are uncorrelated.The converse in not always true.
Next,we consider n-element X and m-element Y as column vectors.

What is the Covariance and Correlation in this case?

- Covariance

$$cov(X,Y) := E[(X-\bar{x})(Y-\bar{y})^T]$$

- Correlation $[E(XY^T)] = \begin{bmatrix} E(X_1Y_1) & \dots & E(X_1Y_m) \\ \vdots & & \vdots \\ E(X_nY_1) & \dots & E(X_nY_m) \end{bmatrix}$

If two random variables $x_icovariatestrongly, knowingaboutX_itellsusalotaboutX_2$.
If covariance is 0,knowing $X_itellsusnothingaboutX_2(theconditionalandmarginaldistrib$
Some important properties

$$E[Y] = aE[X] + b$$

if Y is a linear function
Conditional Expectation

$$E[X|Y] = \sum_x xp_{X|Y}(x|y)$$

$$E[X|Y] = \int_{-\infty}^{\infty} x f_{X|Y}(x|y)$$

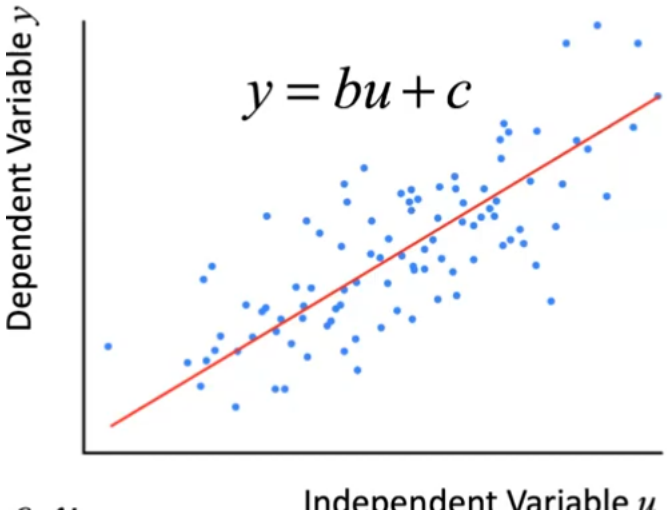
Note that the conditional expectation provides minimum variance
Bayes'rule for contiuous RV

$$f_{X|Y]} = \frac{f_X(x) f_{Y|X}(y|x)}{f_Y(y)}$$

2. Least squares Estimation for Deterministic system

2.1 Regression Analysis

A statistical method for exammining relationship between multiple Variable.In particular, it is to analyze the influence of independent variable upon dependent variables.



y=bu+c

y:Dependent variable
u:Independent variable
b,c:Parameter to estimate

2.2 Linear Regression

In general,Consider a linear homogeneous equation

$$y = b_1 u_1 + b_2 U_2 + \ldots + b_m u_m \quad (1)$$

Where:

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{b_1} \\ \vdots \\ \boldsymbol{b_m} \end{bmatrix} \boldsymbol{\epsilon R^{n\times 1}} : Parametervector$$

$$\boldsymbol{\varphi} = \begin{bmatrix} \boldsymbol{u_1} \\ \vdots \\ \boldsymbol{u_m} \end{bmatrix} \boldsymbol{\epsilon R^{m\times 1}} \quad : Regressor$$

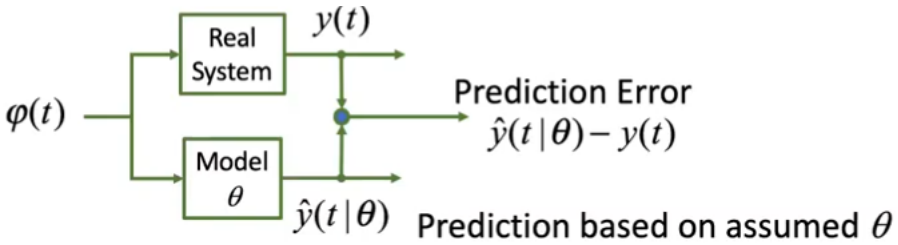
Dependent Variable y is given as an inner product between θ and φ:

$$y = \theta^T \varphi (= \varphi^T \theta) : LinearRegressior$$

2.3 Leat Squares Estimate(LSE)

For finding parameter from Data, y(t) and φ

- prediction - Error Formalism



- Mean Squad Error:

$$V_N(\theta) = \frac{1}{N}\sum_{t-1}^{N}(\hat{y}(t|\theta) - y(t))^2$$

- Least squares estimate(LSE) provided the parameter vector that minimizes the above Mean squard Error:

$$\hat{\theta}^{\mathbf{LS}} = \mathbf{argmin} V_{\mathbf{N}}(\theta)$$

min θ

2.4 Prediction Error formalism

Broadly used in estimation, system identification, and machine learning

Solution: The necessary conditions for V_N to take a minimum:

$\frac{\mathrm{d}\mathbf{V_N}(\theta)}{\mathrm{d}\theta} = \mathbf{0}$: Differentiation of a scalar function V_N with respect to vector θ

$\frac{\partial \mathbf{V_N}(\theta)}{\partial \theta_\mathbf{i}} = \frac{\mathbf{1}}{\mathbf{N}}\sum_{\mathbf{t=1}}^{\mathbf{N}} \frac{\mathrm{d}\mathbf{x_t^2}}{\mathrm{d}\mathbf{x_t}} \frac{\partial \mathbf{x_t}}{\partial \theta_\mathbf{i}} = \frac{\mathbf{2}}{\mathbf{N}}\sum_{\mathbf{t-1}}^{\mathbf{N}} \mathbf{x_t} \frac{\partial \hat{\mathbf{y}}(\mathbf{t}|\theta)}{\partial \theta_\mathbf{i}} = \frac{\mathbf{2}}{\mathbf{N}}\sum_{\mathbf{t-1}}^{\mathbf{N}} \mathbf{x_t}\varphi_\mathbf{i}(\mathbf{t})$

$V_N(\theta) = \frac{1}{N}\sum_{t-1}^{N} X_t^2$ where $x_t = \hat{y}(t) - y(t)$

$\hat{\mathbf{y}}(\mathbf{t}|\theta) = \theta_\mathbf{1}\varphi_\mathbf{1} + \ldots + \theta_\mathbf{i}\varphi_\mathbf{i} + \ldots + \theta_\mathbf{m}\varphi_\mathbf{m}$

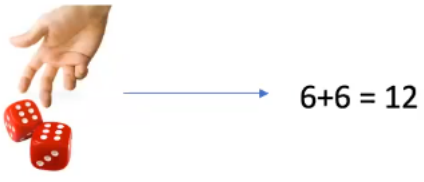
$$\Rightarrow \begin{pmatrix} \frac{\partial V_N(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial V_N(\theta)}{\partial \theta_i} \\ \vdots \\ \frac{\partial V_N(\theta)}{\partial \theta_m} \end{pmatrix} = 2 \frac{}{N\sum_{t-1}^{N}(\hat{y}(t|\theta) - y(t))} \begin{pmatrix} \varphi_1(t) \\ \vdots \\ \varphi_i(t) \\ \vdots \\ \varphi_m(t) \end{pmatrix}$$

3. Random_Variable and Random_Process

3.1 Probability and Random Variable

3.1.1 Random Variable

Random variable is a function that maps a random event to a numerical value.



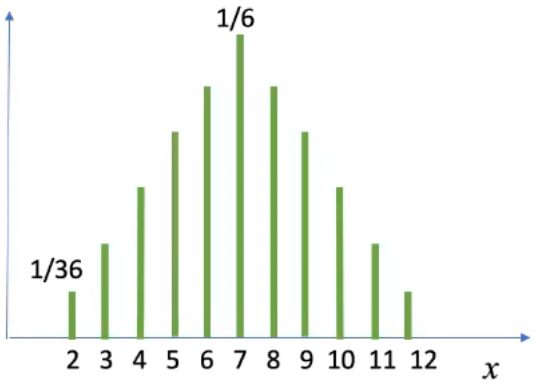
Event

Random event: ω $\xrightarrow{X}$ Number

Random Variable Capital $X = x_i$ Instantiation Lower Case

Probability Mass Function

$$\mathbf{P_X(x_i) = Pr(X = x_i)}$$



3.1.2 Cumulative Distribution Function (CDF)

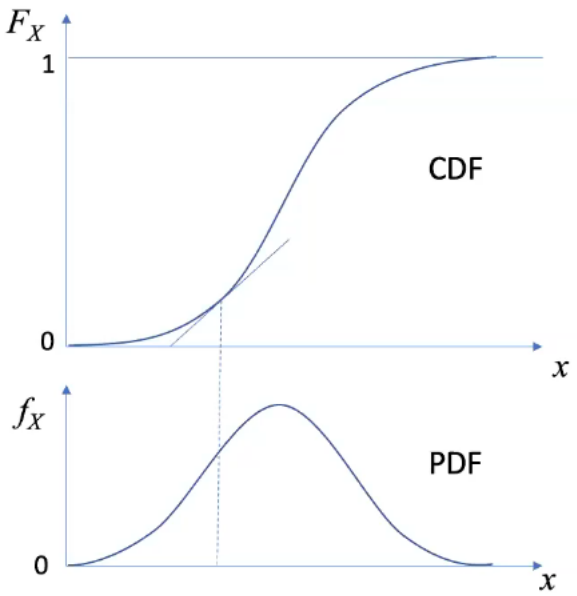
$$\mathbf{P_X(x_i) = Pr(X = x_i)}$$

Note that for continuous X,Pr(X=x)=0

3.1.3 Probability Density Function (PDF)

$$\mathbf{f_X(x) = \frac{dF_X(x)}{dx}}$$

$$\mathbf{F_X(x) = \int_{-\infty}^{x} f_X(z)\,dz = \int_{-\infty}^{\infty} f_X(z)\,dz = 1 \quad f_X(x) >= 0}$$



3.1.4 Joint Probability

Discrete: $p_{XY}(x_i, y_i) = Pr(X = x_i \& Y = y_i)$ simultaneously
Continuous: $f_X Y(x, y)\Delta x \Delta y = Pr(x + \Delta x \& y \leq Y + \Delta y)$

3.1.5 Independent Random Variables

$$f_{XY}(x, y) = f_X(x) f_Y(y), \quad \bigvee x, \bigvee y for all x and y.$$

3.1.6 Conditional Probability Density

$$f_{X|Y}(x|y) = \frac{f_{XY}(x}{f_Y(y)}$$

If X and Y are independent,

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

3.1.7 Bayes Rule

$$f_{Y|X}(y|x) = \frac{f_{XY}(x, y)}{f_X(x)}$$

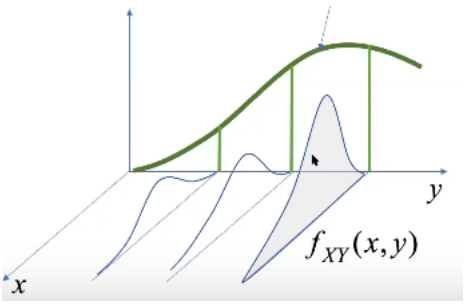
$$f_{XY}(x, y) = f_{Y|X}(Y|X) f_x(x) = f_{X|Y}(x|y) f_Y(y)$$

$$\Rightarrow \mathrm{f}_{X|Y}(x|y) = \frac{f_{Y|X}(y|x) f_X(x)}{f_Y(y)}$$

3.1.8 Marginal Probability

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

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



3.1.9 Expectation

Expected value of X , Synonym:mean,average

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

$$Discrete : \mathbf{E[X]} = \sum_{\mathbf{i}}^{\mathbf{N}} \mathbf{x_i p_i}$$

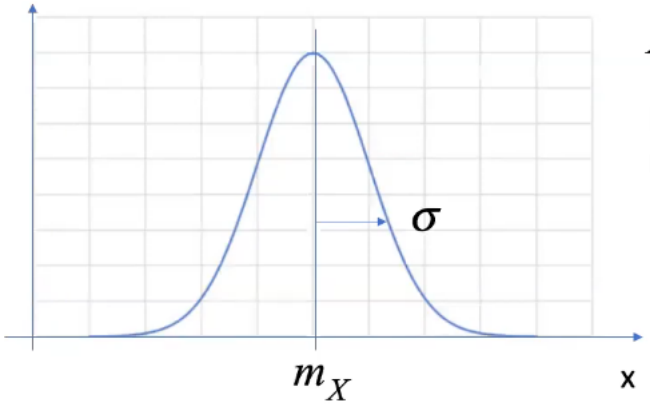
3.1.10 Variance

$$var[X] = E[)X - E[X])^2] = E[X^2 - 2XE[X] + (E[X])^2$$
$$= E[X^2 - 2XE[X] + (E[X])^2$$
$$\Rightarrow var[X] = E[X^2] - (E[X])^2$$

3.1.11 Moment

K-th moment $E[X^k] = \int_{-\infty}^{\infty} x^k f_X(x)\,dx$ k=1,2,3,...

3.1.12 Normal(Gaussian) Distribution



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

Random variable X has a normal distributio
mean m_X and variance σ^2

$$\sigma^2 = \int_{-\infty}^{\infty} (x - m_X)^2 f_X(x)$$

PDF

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

Mean and variance completely characterize the distribution.

3.1.13 Correlation

the expectation of the product ot two random variable,X and Y,is

called"Correlation".

$$\mathbf{E[XY]} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbf{xyf_{XY}(x,y)\,dxdy}$$

If X and Y are independent,

$$E[XY] = \int_{-\infty}^{\infty} xf_x(x)dx \int_{-\infty}^{\infty} yf_Y(y)\,dy = E[X]E[Y]$$

Note:Although Correlation is zero,the two random varible are not necessarily independent.

3.1.14 Orthogonality

If correlation is zero, E[XY]=0, X and Y are said to be "Orthogonal".

3.1.15 Covariance

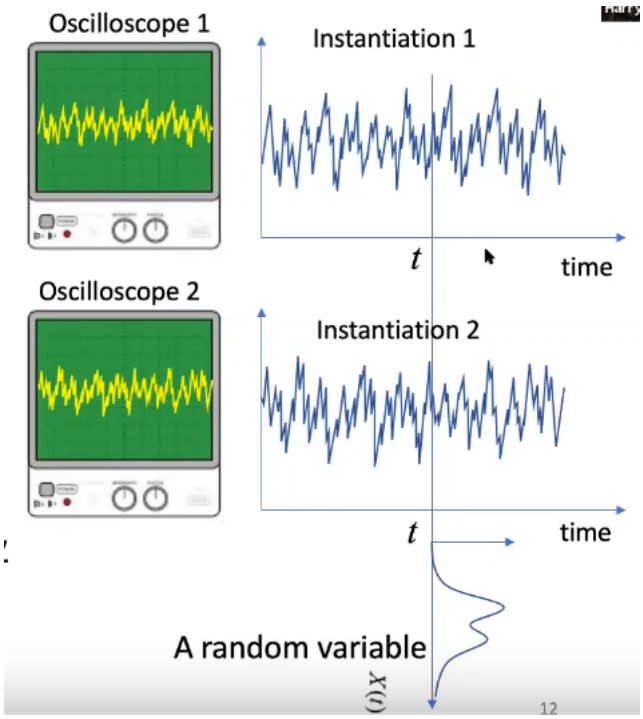
$$\mathbf{cov[X,Y]} = \mathbf{E[(X - E[X])(Y - E[Y])]}$$

3.1.16 Correlation Coefficient

$$\rho_{\mathbf{XY}} = \frac{\mathbf{cov[X,Y]}}{\sqrt{\mathbf{var[X]}}\sqrt{\mathbf{var[Y]}}} = \mathbf{E}[\frac{\mathbf{X - m_x}}{\sigma_\mathbf{x}}\frac{\mathbf{Y - m_y}}{\sigma_\mathbf{Y}}] \quad -1 \leq \rho_{XY} \leq 1$$

3.2 Random Process

- A random process is a family(ensemble) of time funtion having a probability measure.
- Consider a set of identical oscilloscopes measuring Ground Noise.
- Each oscilloscope shows a pacticular noise waveform coming from the same source of random process, that is to say, each waveform is an instantiation.
- Collecting voltage values displayed in all the oscilloscope ,we can construct a probability distribution, as shown below.
- At each time the output of an oscilloscope is therefore a random variabel, X(t).
- The random variable, X(t),$-\infty < t < \infty$ are collectively called a Random Proccess.



Quantification of a random process:

- First-Order Density $f_{X(t)}(x)$

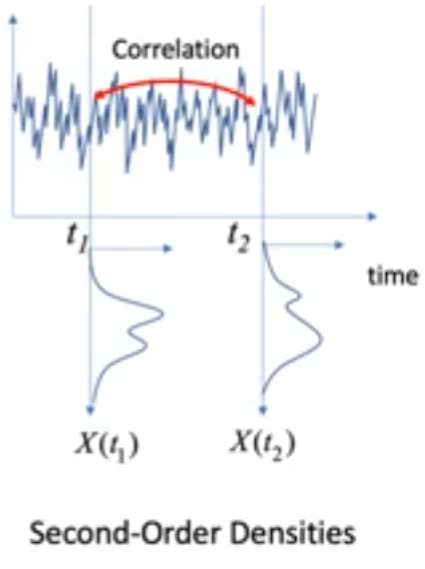
2.2.1 Auto-Correlation

Quantification of a random process with correlation

- Let us quantify how two random variable at two different times of a random process are correlated to each other.
- Let $X(t_1)$ and $X(t_2)$ be two random variables at time t_1 and t_2.
- Let $f_{x_1x_2}(x_1, x_2$ be the joint probability of the two random variables.
- The correlation between the two is given by:

$$\boldsymbol{R_{XX}(t_1, t_2) = E[X(t_1)X(t_2)] = \int \int_{-\infty}^{\infty} x_1 x_2 f_{x_1x_2}(x_1, x_2) dx_1 dx_2}$$

This is called **Auto Correlation**,since it is from the same random process.



Second-Order Densities

2.2.2 Wide-Sense Stationary Process

A random process is called a Wide-Sense Stationary Process, If the following two conditions are met:

- The mean value does not depend on time;the mean is uniform and constant throughout the process:

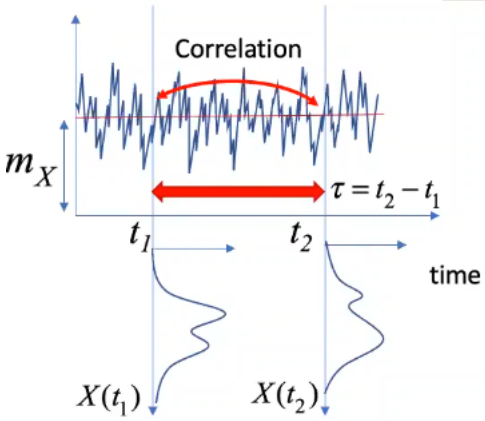
$$\boldsymbol{E[X(t)] = m_X;\ \forall t}$$

- The Auto-Correlation depends only on the time interval,$\tau = t_2 - t_1$, rather than the specific times

$$R_{XX}(t_1, t_2) = R_{XX}(\tau) = E[X(t+\tau)X(t)];\ \forall t$$

Note that the auto-correlation of a wide-sense stationary process is an even function.

$$\boldsymbol{R_{XX}(-\tau) = R_{XX}(\tau)}$$



Auto-Covariance: $C_{XX}(t_1, t_2) = E[(X(t_1) - m_X(t_1))(X(t_2) - m_X(t_2))]$

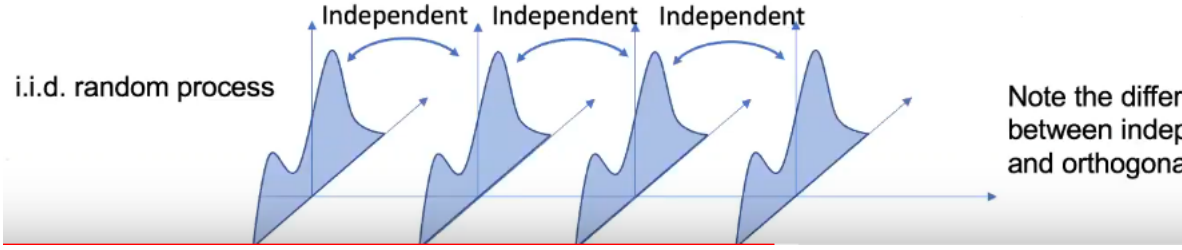
$$= R_{XX}(t1_1, t_2) - m_x(t_1)m_x(t_2)$$

2.2.3 Independent and Identically Distributed(i.i.d. IID, iid) Random Variables and Processes

- Consider two random variables X and Y,with probability desities $f_X(x)$ and $f_Y(y)$.
- Random Variables X and Y are called Independent and Indentically Distributed(i.i.d)if the following two conditions are met:

$$f_X(x) = f_Y(x) \quad \forall x, \forall y$$

- i.i.d is a convenient assumption widely used in statistics.Often it reflects samples taken from the same source.
- This concept can be easily extended to more than two random variable and random processes .

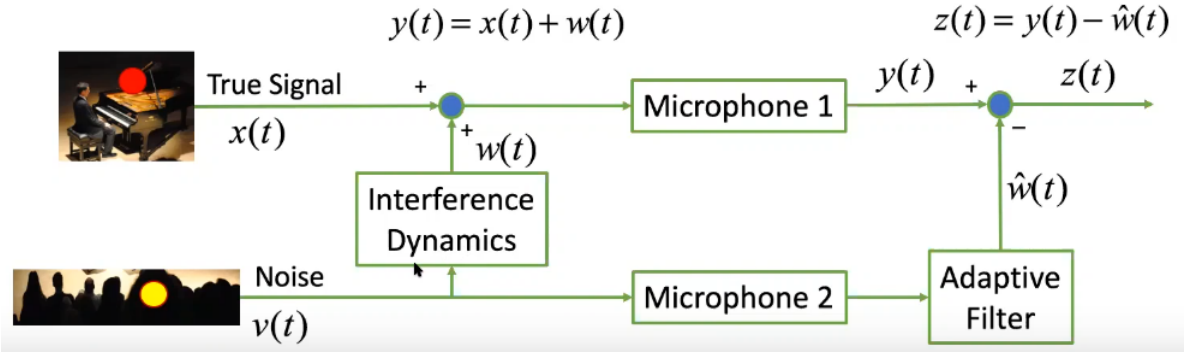


2.2.4 Adaptive Noise Cancellation: An Appliction

- Before concluding this chapter,we apply the theory of random variables and random processes to practical problem.
- Active noise cancellation is a technique to measure the background acoustic noise and substract it from the main signal so that the latter is not corrupted with the background noise.

2.2.5 Active Noise Cancellation

- The background noise v(t) propagates through an unknown dynamic process adn arrives at Microphone 1, where the true signal x(t) is mixed with the interfering noise w(t).
- The background noise picked up with Microphone 2 is fed to Adaptive Filter,where the interfering noise is recovered.
- The predicted interfering noise $\hat{w}(t)$ is subtracted from the output of Microphone 1,y(t),to recover the true signal.



We assume a Finite impulse Model for the interfering Dynamics.

$$w(t) = b_1 v(t-1) + b_2 v(t-2) + \ldots + b_m v(t-m) \; -> \; w(t) = \theta^T \phi(t) : Alinearregression$$

Parameters to estimate:

$\theta = \begin{Bmatrix} b_1 \\ \vdots \\ b_m \end{Bmatrix}$ Regressor: $\phi(t) = \{v(t-1)//\ldots//v(t-m)\}$ **Assumption:**True signal x(t) and background noise v(t) are uncorrelated random processes,

$$\boldsymbol{E[x(t)v(t-\tau)] = 0, \quad \forall t, \forall \tau}$$

Problem: Estimate paramate vector θ in real-time,so that the noise may be most suppressed.
Solution: Consider the recovered output z(t),which depends on the parameter values:

$$\boldsymbol{z(t|\hat{\theta}) = y(t) - \hat{w}(t|\hat{\theta})}$$

The mean squared signal strength of z(t—$\hat{\theta}$)

$$\boldsymbol{E[(z(t|\hat{\theta}))^2] = E[(y(t) - \hat{w}(t|\hat{\theta}))^2] = E[(x(t) + w(t) - \hat{w}(t|\hat{\theta}))^2]. \quad = E[(x(t))^2]}$$

Examine

$$= b_1 E[x(t)v(t-1)] + \ldots + b_m E[x(t)v(t-m)] = 0$$

since the true signal and background noise are uncorrelated: E[x(t)v(t-τ)]=0, $\forall t, \forall \tau$

Similarly, E[x(t)(t)]=E[x(t){$\hat{b_1}$v(t-1)+...+$v(t-m)$}] =E[x(t)v(t-1)]+...+$\hat{b_m}E[x(t)v(t-m)] = 0$

$$E[(z(t\,|\,\hat{\theta}))^2] = E[(x(t))^2] + 2E[x(t)w(t)] - 2E[x(t)\hat{w}(t\,|\,\hat{\theta})] + E[(w(t) - \hat{w}(t\,|\,\hat{\theta}))^2]$$

This term does not depend on parameter θ.

$E[x(t)w(t)] = 0$

$E[x(t)\hat{w}(t)] = 0$

This can be treated as a prediction error.

$$\theta^o = \arg\min_{\hat{\theta}} E[(w(t) - \hat{w}(t\,|\,\hat{\theta}))^2]$$

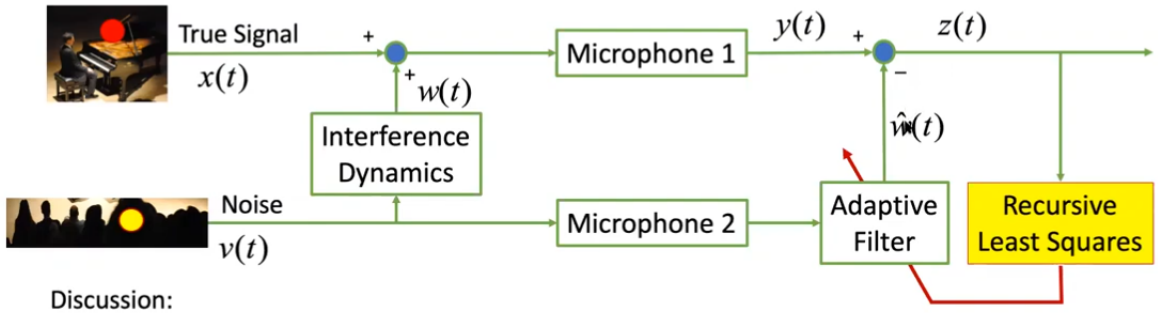
We want to minimize this squared error in predicting the interfering noise.

Treating w(t)-$\hat{w}(t|\hat{\theta})$ as a prediction error and replacing it by z(t—$\hat{\theta}$),we can apply the Recursive Least Squares algorithm to estimate the parameters involed in the interfering dynamics (FIR model) in real-time.Using forgetting factor α,

$\hat{\boldsymbol{\theta}}(\boldsymbol{t}) = \hat{\boldsymbol{\theta}}(\boldsymbol{t-1}) + \frac{\boldsymbol{P_{t-1}\phi(t)}}{\boldsymbol{\alpha + \phi_T(t) P_{t-1}\phi(t)}}[\boldsymbol{y(t)} - \hat{\boldsymbol{y}}(\boldsymbol{t}|\hat{\boldsymbol{\theta}}(\boldsymbol{t-1}))]$

$=> \boldsymbol{P_t} = \frac{\boldsymbol{1}}{\boldsymbol{\alpha}}\left[\boldsymbol{P_{t-1}} - \frac{\boldsymbol{P_{t-1}\phi(t)\phi^T(t)P_{t-1}}}{\boldsymbol{\alpha + \phi^T(t)P_{t-1}\phi(t)}}\right]$

Active Noise Cancellation using RLS and Orthogonality



Discussion:

1. What if the true signal x(t) is picked up by Microphone 2, too?
2. Does $\hat{b}_i$ correlate with x(t)? If so, it cannot be factored out.

Answer:

Context-Oriented Project #1 is on a related topic.
You should discuss it in your study group.

4. Principal Component Regression

4.1 Rank-Deficit Data Matrix

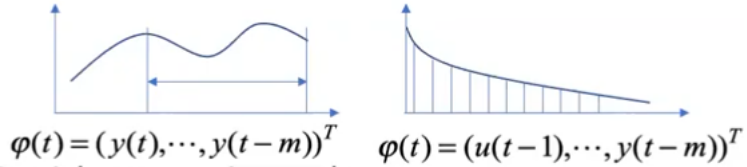
In previous chapters on Least Squares Estimate and Recursive Least Squares, we assumed that

$$\sum_{i=1}^{t} \boldsymbol{\phi(i)\phi^T(i) = Non - sigulare, P_0 = Positive - Definite}$$

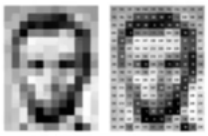
This assumption may not be valid for:

- High input dimension:m¿¿1

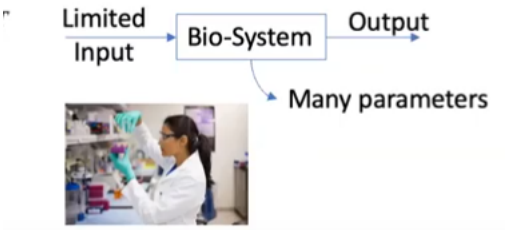
- Temporal case, e.g. a wide time window, slowly-decaying FIR model



- Spatial case, e.g. image data


Pixels: m = (# of row) x (# of column)

- Limited sample size:N¡m
  - Difficulty in obtaining a large number of data under consistent conditions, e.g. clinical trial data, biological experiment



- When we do not know which input signals are important,we in clude as many potentiallly important inputs as possible.

Approach

1.Use the pseudo-inverse
2.Regularization

- Ridge Regression
- Principal Component Regression
- Partial Least Squares Regression

4.2 Quick Math: More-Penrose Pseudo-inverse

Cosider a linear simultaneous equation: Ax=b where A:m x n

- If there is no solution to Ax=b,then the pseudoinverse of A,denoted $\mathbf{A}^{\#}$, minimizes the squared error $|Ax - b|^2$ abd is given by

$$\boldsymbol{A^{\#} = (A^T A)^{-1} A^T}$$

with this , the solution is $x^{=}A^{\#b}$
This corresponds to the Least Squares solutions:

$$\boldsymbol{\theta = \left(\sum_{i=1}^{t} \phi(i)\phi^T(i) \right)^{-1} \sum_{i=1}^{t} \phi(i) y(i)}$$

- If there are many (infinite)solutions to Ax=b,e.g. n¿m, the n pseudoinverse $A^{\#}$ provides the solution that minimizes the square norm of vector x.

$$min|x|^2 \ \ subjecttoAx = b$$

$$\hat{x} = A^{\#}b \;\; A\hat{x} = b \; and \; min|x|^2$$

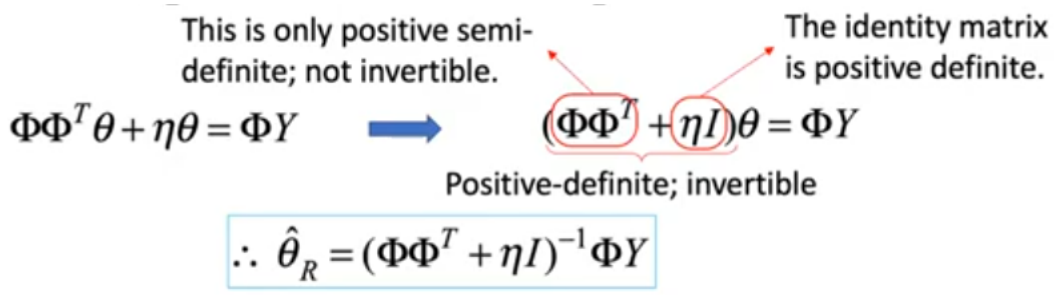
Definition:
Given a matrix $A \in C^m$,the matrix $A^{\in C^{m\times n}}$ that satisfies the following four coditions is called the More Penrose Pseudoinverse:
1.$AA^{\#}A = A$
2.$A^{\#}AA^{\#} = A^{\#}$
3.$(AA^{\#})^* = AA^{\#}$
4.$(A^{\#}A)^* = A^{\#}A$
Where $(X)^*$ is conjugate transpose of X.
A pseudoinverse is unique.

4.3 Ridge Regularization

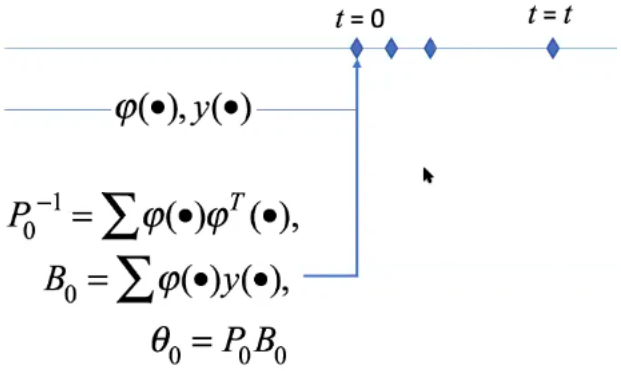
Consider a cost funtional (penalty): $\mathbf{V_R}(\theta) = |\mathbf{Y} - \mathbf{\Phi^T}\theta|^2 + \eta|\mathbf{\Theta}|^2$ where $\eta > 0$ is a weight that determines the trade-off between the squared prediction error and the magnitude of the paramete vector.
Find the parameter that minimizes it: $\hat{\Theta}_R = argmin V_R(\Theta)$

$$\frac{dV_R(\Theta)}{d\Theta} = 0 => \frac{d}{d\Theta}\left[(Y - \Phi^T\Theta)^T(Y - \Phi^T\Theta) + \eta\Theta^T\Theta\right]$$

$$= \frac{d}{d\Theta}\left[Y^TY - 2Y^T\Phi^T\Theta + \Theta^T\Phi\Phi^T\Theta + \eta\Theta^T\Theta\right] = 0$$



4.4 Initial Conditions for Recusive Least Squares can be viewed as a type of regularization



- Suppose that before t=0, there were some data $\Phi(\bullet), y(\bullet)$
- $P_0, B_0, and\theta_0 werecomputedhasedonthesedata$
- Consider the following cost funtion J_0.The parameter vector θ that minimizes J_0 is the initial condition θ_0

$$\boldsymbol{J_0} = \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta_0})^T\boldsymbol{P_0^{-1}}(\boldsymbol{\theta} - \boldsymbol{\theta_0})$$

- As shown in Lecture Notes Chapter2 Section 2.4,We can prove that the recursive leat squares algorithm, starting with the initial condition θ_0 and P_0,is the optimal

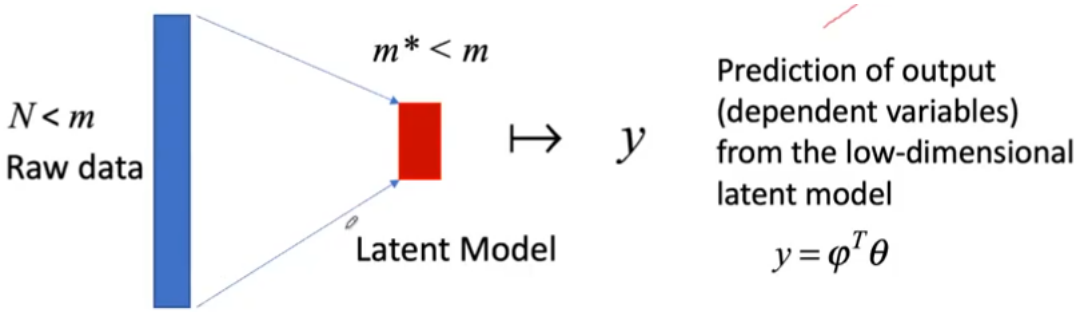
value that minimizes the following cost funtion.

$$\boldsymbol{J_t = \frac{1}{2}\sum_{i=1}^{t}(y(i) - \phi^T(i).\theta)^2 + \frac{1}{2}(\theta - \theta_0)^T P_0^{-1}(\theta - \theta_0)}$$

- The second term above can be viewed as a regularization term.

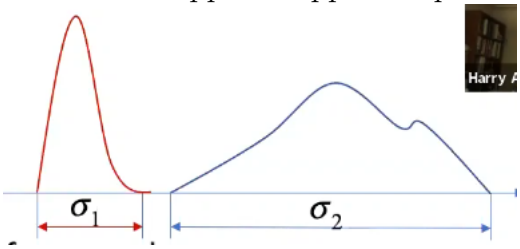
4.5 Statistical Multivariate Analysis

- Examine relationships among multiple variables in a high dimensional space;
- Joint behaviors of multiple variables may in dicates some redunancy, and two variables may be linearly correlated:collinear.
- We are interested in extracting succinct, significant variable from high dimensional raw data;
- **Latent** Variable Method is to find hidden or encapsulated variables in the raw data that represent the raw data in a low dimensional space:Latent Model.
- Predict the output from the low-dimensional latent model
  - Principal Components Analysis and Regrssion
  - Partial Least Squares Regression



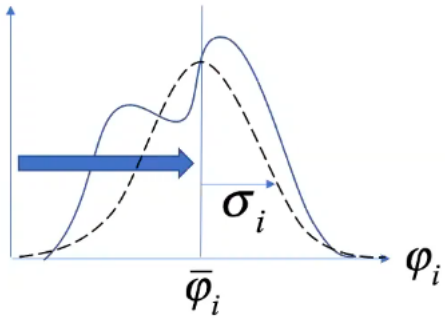
4.6 Pre-Processing

- A regressor vector contains multiple variables having different units and scales; Example:$\phi = (10kg, 5m/s, 2cm)^T$.
- To make an apple-to-apple comparison,the variables must be normalized.



♦$Mean - centering$
$Shifttheorigintothemean\bar{\phi}$

$$\varphi_i \mapsto \varphi_i - \bar{\varphi}_i$$



♦$Normalization with some reference value \boldsymbol{x_i} = \frac{\phi_i - \bar{\phi}_i}{\boldsymbol{\sigma_i}}$
. reference value: σ_i

- Standard deviation
- Max-Min (dynamic range)

Input Data Matrix
X=[x(1) $\ldots x(N)] \in R^{mxN}$ Output Data Matrix
Y=$\begin{Bmatrix} y(1) \\ \ldots \\ y(N) \end{Bmatrix} \in R^{Nx1}$

5. **Principal Component Regression:A Multi-Input,Single-Output Case**

- Examine how input data are distributed in the m-dim space and reduce the space to a low dimensional space;
- The distribution can be characterized with the covariance of preprocessed input data:

$$X = \{x(1) \ldots x(N)\} \in R^{m \times N}$$

Recall we have defined Covariance of two scalar random variables,X and Y,as

$$\mathbf{cov}(\mathbf{X}, \mathbf{Y}) = \mathbf{E}[(\mathbf{X} - \mathbf{E}[\mathbf{X}])(\mathbf{Y} - \mathbf{E}[\mathbf{Y}])]$$

Covariance of a vector consists of covariances of all the vector components.

$$\boldsymbol{C = cov}\left[\begin{Bmatrix} \boldsymbol{x_1} \\ \vdots \\ \boldsymbol{x_m} \end{Bmatrix} \begin{Bmatrix} \boldsymbol{x_1} & \vdots & \boldsymbol{x_m} \end{Bmatrix}\right] = \begin{bmatrix} \boldsymbol{E[x_1^2]} & \boldsymbol{\ldots} & \boldsymbol{E[x_1 x_m]} \\ \vdots & \ddots & \vdots \\ \boldsymbol{E[x_m x_1]} & \boldsymbol{\ldots} & \boldsymbol{E[x_m^2]} \end{bmatrix} = \boldsymbol{E[X X^T]}$$

Note that input data have been mean-centered.

- What is the physical sense of the covariance matrix?
- How can we use it for latent modeling ?
  Recap Examine the strength of signals in the direction of unit vector v.

$$\boldsymbol{J = \frac{1}{N} \sum_{i=1}^{N} |v^T x(i)|^2 = \frac{1}{N} v^T (\sum x(i).x^T(i)) v}$$

$$\boldsymbol{= \frac{1}{N} v^T [x(1) \ldots x(N)] \begin{bmatrix} x^T(1) \\ \vdots \\ x^T(N) \end{bmatrix} v = \frac{1}{N} v^T X X^T v}$$

The j-k component of matrix XX^T $\frac{1}{N} x_j(1)x_k(1) + \ldots + x_j(N)x_k(N) \cong E[x_f x_k]$ $\therefore$ $J \cong v^T C v$
Find the strongest direction of signal in the input space: an eigenvalue problem.
$\boldsymbol{v = argmax_v (v^T X X^T v - \lambda(v^T v - 1)} \Longrightarrow \mathrm{XX}^T \boldsymbol{v = \lambda v}$
Eigenvalues. $\lambda_m ax = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \geq 0$
We are considering the case where $XX^T \cong C$: singular.
Suppose that the last(m-m*)eigenvalues are zero.
$\lambda_m ax = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \geq 0 \quad \lambda = 0, \lambda_{m-1} = 0, \lambda_{m+1} = 0$

$$XX^T = \begin{bmatrix} v_1 & \dots & v_m \end{bmatrix} \begin{bmatrix} \lambda_1 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & & \lambda_{m*} & & 0 \\ \vdots & & & 0 & \\ 0 & \dots & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_m^T \end{bmatrix}$$

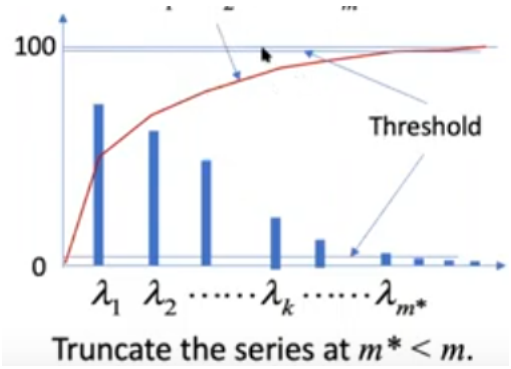
$$\therefore \boldsymbol{XX^T = \lambda_1 v_1 v_1^T + \dots + \lambda_{m*} v_{m*} v_{m*}^T + 0 + \dots + 0}$$

$The first$m* components can generate and fully explain the data.

In practice,some eigenvalues are small but not exactly zero.We can truncate them with a threshold.

Percent Accuracy: a measure for truncation

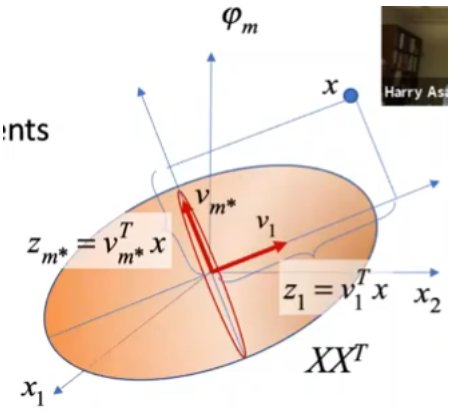
$$\boldsymbol{\mu = \frac{\lambda_1 + \lambda_2 + \dots + \lambda_k}{\lambda_1 + \lambda_2 + \dots + \lambda_m} \times 100\%}$$



Step1: Reduce the input space and represent if with Pricipal Components $\therefore XX^T = \lambda_1 v_1 v_1^T + \dots + \lambda_{m*} v_{m*} v_{m*}^T + 0 + \dots + 0$

. Define $m^* < m$ Latent Variables using eigenvectors associated with the top m^* most significant eigenvalues

$$\boldsymbol{z_1 = V_1^T x, z_2 = v_2^T x, ..., z_{m*} = v_{m^*}^T x}$$



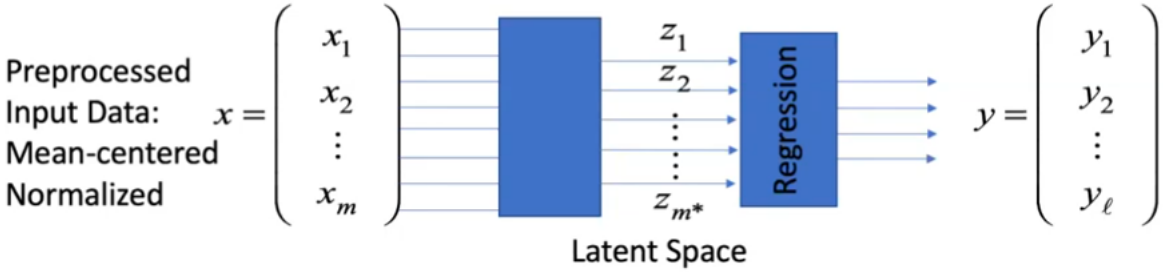
Step2: Construct a low-dimensional regression on the Principal Components

$$\boldsymbol{\hat{y} = b_1 z_1 + b_2 z_2 + \dots + b_{m*} z_{m*} \quad m^* < m}$$

$$\boldsymbol{\theta = \begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m^*} \end{Bmatrix} = \left[\sum_i \begin{Bmatrix} z_1(i) \\ \vdots \\ z_{m^*}(i) \end{Bmatrix} \begin{Bmatrix} z_1(i) & \dots & z_{m^*}(i) \end{Bmatrix} \right]^{-1} \left[\sum_i \begin{Bmatrix} z_1(i) \\ \vdots \\ z_{m^*}(i) \end{Bmatrix} y(i) \right]}$$

6. Partial Least Squares Regression

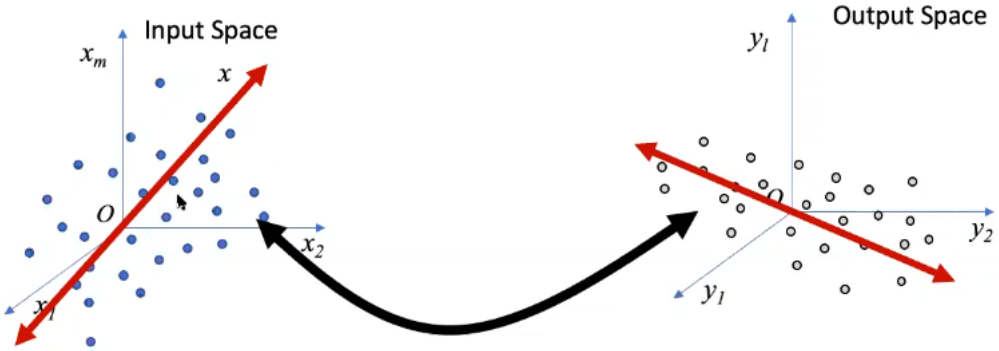
6.1 Latent Modeling



- Principal Component Regression: Characterizes the input data space,reduces the input dimension based on principal component Analysis, and regresses on pricipal components.
- Caveat! Small principal components,which are ignored,may be highly correlated with outputs.Those components must not be neglected.
- Components having significant correlation with outputs must be involved in the latent space.
- This requires to analyze both input and output spaces,rather than characterizing the input space alone.
- Multiple Outputs:Unlike single output regressions, we often need to estimate multiple outputs,which may be correlated
- This lecture will discuss the latent space modeling based on input-output correlation analysis.

6.2 The Core Algorithm of Multi-Input, Multi-Output Partial Least Squares Regression

Partial Least Squares Regression is a latent modeling method for predicting a set of outputs in relation to a reduced order inputs.The basic idea is to find a low-dimensional set of input space variables that is most correlated with a given set of output data.It is to analyze data in both input space and output space.

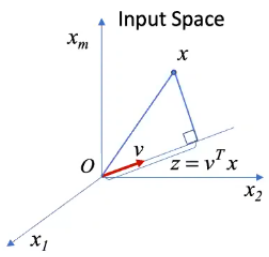


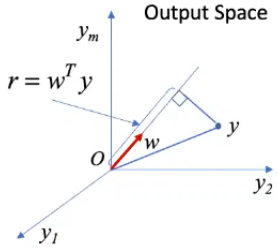
- **Step1:**Find the directions of a pair of unit vector, $v \in \mathrm{Re}^m$ in the input space and $\in \mathrm{Re}^l$ in the output space,that maximizes the correlation between the projection of input vector onto the unit vector, $z = v^T x$, and that of the output vector,$r = w^T y$.

$$\max_{v,w} E[z \cdot r] = \max_{v,w} E[v^T x \cdot w^T y]$$
$$= \max_{v,w} v^T \underbrace{E[x\,y^T]}_{C_{XY}} w$$

where $|v| = 1, |w| = 1$

Covariance of mean-centered random variables x and y.





$z = v^T x$ is called the **score** of input x with respect to v,and $r = w^T y$ is called the **score** of output y with respect to w.
Problem:

$$\max_{v,w} \boldsymbol{v^T C_X Y w} \quad Subjectto |v| = 1, |w| = 1$$

Solution: Using Lagrange's multiplers,

$$(\boldsymbol{v^o, w^o}) = \boldsymbol{arg} \max_{v,w} \left\{ \boldsymbol{v^T C_{XY} w} - \frac{1}{2}\boldsymbol{\lambda_v}(\boldsymbol{v^T v} - 1) - \frac{1}{2}\boldsymbol{\lambda_w}(\boldsymbol{w^T w} - 1) \right\}$$

The necessary conditions for v and w to maximize the correlation are:
$\frac{\partial}{\partial v} = 0 \Rightarrow \boldsymbol{C_{XY} w - \lambda_v v = 0}$ **(1)**
$\frac{\partial}{\partial w} = 0 \Rightarrow \boldsymbol{(C_{XY})^T v - \lambda_v w = 0}$ **(2)**
Note that by definition:$(C_{XY})^T = C_{YX}$
From (2), $\mathbf{w} = \frac{1}{\lambda_v}\mathbf{C_{YX}v}$. Substituting this in (1) yields. $\mathbf{C_{XY}C_{YX}v} = \lambda_\mathbf{v}\lambda_\mathbf{w}\mathbf{v}$
This implies that vector v is an eigenvector of matrix $C_{XY}C_{YX}$
Similarly,from(1) $v = \frac{1}{\lambda_v} C_{XY} w$.Substituting this into (2) yields :
$\boldsymbol{C_{YX}C_{XY}w = \lambda_v\lambda_w w}$
. The implies that vector w is an eigenvector of matrix $C_{YX}C_{XY}$
Transpose of a Covariance Matrix

$$(\mathbf{C_{XY}})^\mathbf{T} = \begin{pmatrix} E[x_1y_1] & \dots & E[x_1y_l] \\ \vdots & \ddots & \vdots \\ E[x_my_1] & \dots & E[x_my_l] \end{pmatrix}^T = \begin{pmatrix} E[x_1y_1] & \dots & E[x_my_1] \\ \vdots & \ddots & \vdots \\ E[x_1y_l] & \dots & E[x_my_l] \end{pmatrix}$$
$$= \begin{pmatrix} E[x_1y_1] & \dots & E[y_lx_m] \\ \vdots & \ddots & \vdots \\ E[y_lx_1] & \dots & E[y_lx_m] \end{pmatrix} = E\left(\begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix} \begin{pmatrix} x_1 & \dots & x_m \end{pmatrix} \right) = \mathbf{C_{YX}}$$

Or,simply

$$\left(\begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \begin{pmatrix} y_1 & \dots & y_l \end{pmatrix}^T \right) = \begin{pmatrix} y_1 & \dots & y_l \end{pmatrix}^T \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix} \begin{pmatrix} x_1 & \dots & x_m \end{pmatrix}$$

$\boldsymbol{C_{XY}C_{YX}v = \lambda_v\lambda_w v} \quad \boldsymbol{C_{YX}C_{XY}w = \lambda_v\lambda_w w}$
Note that in both cases the eigenvalue is the same: $\lambda_v\lambda_w$
We can show that $\lambda_v = \lambda_w$
Pre-multiplying v^T to (1), $C_{XY}w - \lambda_v v = 0$

$$v^T C_{XY} w - \lambda_v v^T v = 0 \quad \therefore \lambda_v = v^T C_{XY} w$$

Pre-multiplying w^T to (2), $(C_{YX})^T v - \lambda_v w = 0$

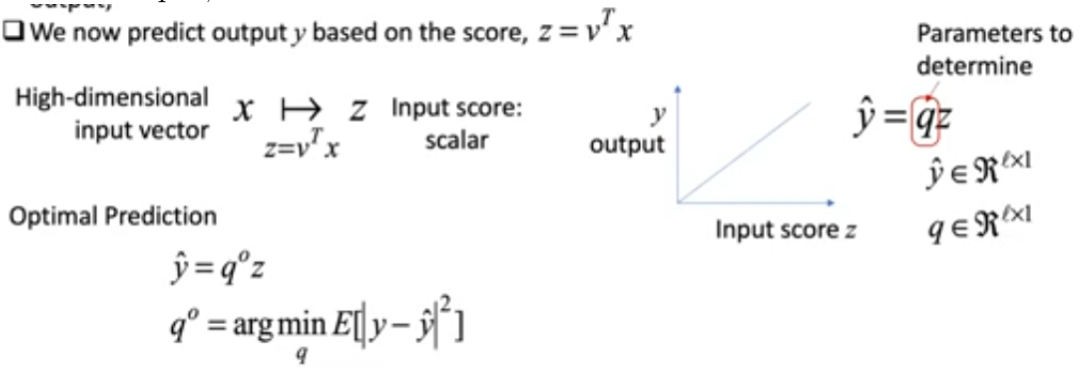
$$w^T (C_{XY})^T v - \lambda_v w^T w = 0 \quad \therefore \lambda_v = w^T (C_{XY})^T v = v^T (C_{XY}) w = \lambda_w$$

$\therefore \lambda_v = \lambda_w$

$C_{XY}C_{YX}$ and $C_{YX}C_{XY}$ have the same eigenvalues. $\lambda_v = \lambda_w = \lambda$

- **Step2:Optimal Prediction with one latent variable**

  ☐ In Step1 we have found the component of the input space that is most correlated with the output;



It is conceivable that this optimal coefficient/parameter vector q^0 is in the same direction as unit vector w. (This is a problem involved in the next assignment.)

Obtaining optimal q^0

$$\hat{\mathbf{y}} = \mathbf{qz} = \mathbf{qv^T x}$$

$$\mathbf{E}[(\mathbf{y} - \hat{\mathbf{y}})^\mathbf{T}(\mathbf{y} - \hat{\mathbf{y}})] = \mathbf{E}[(\mathbf{y} - \mathbf{qv^T x})(\mathbf{y} - \mathbf{qv^T x})]$$

$$= \mathbf{E}[\mathbf{y^T y} - \mathbf{2y^T qv^T x} + \mathbf{x^T vq^T qv^T x}]$$

$$= \mathbf{E}[\mathbf{y^T y} - \mathbf{2v^T xy^T q} + \mathbf{v^T xx^T vq^T q}]$$

$$= \mathbf{E}[\mathbf{y^T y}] - \mathbf{2v^T E}[\mathbf{xy^T}]\mathbf{q} + \mathbf{v^T E}[\mathbf{xx^T}]\mathbf{vq^T q}$$

Note: that **x** and **y** are random variables, **v** and **q** are not.

The necessary conditions for optimality

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{q}} = \mathbf{0} \quad -\mathbf{2E}[\mathbf{yx^T}]\mathbf{v} + \mathbf{2qv^T E}[\mathbf{xx^T}]\mathbf{v} = \mathbf{0}$$

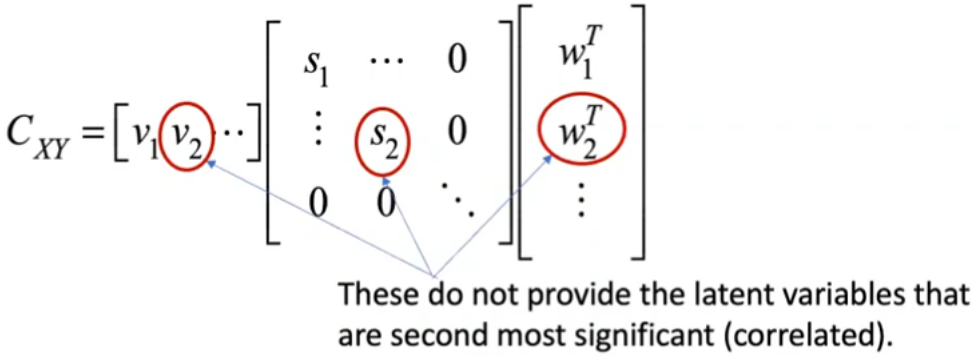
$\therefore \mathbf{q^o} = \frac{\mathbf{C_{YX}v}}{\mathbf{v^T C_{XX}v}}$ This is called **Output Loading Vector.**

- **Step3: Deflation**

  ☐ We have just one set of latent variables associated with the highest correlation between input and output

  ☐ But, and accurate prediction cannot be obtained with just one set of latent variables.Now we want to find the components of the second and the third highest correlation.

  ☐ The singular Value decomposition of the cross-correlation matrix,however,does not directly give the second and the third most significant latent variables .



☐ To overcome this difficulty,we have to go through the procedure called "Deflation".
☐ Predicting output y based on the first set of latent variables, we have used some components involved in the data matrix;
☐ This component of data matrix must not be used for determining the second most significant component;
☐ We have to remove the components already used in the first round prediction, and examine the residual components that have the highest correlation with output.
☐ Output data

ỳ=y - (the component used for the first round output prediction)

where: **ỳ**=Residue , **y**= Original data
☐ Using the output loading vector **q**, the deflated output vector is given by

ỳ=y-zq

☐ The deflation of input data matrix is a bit more involved.Collectively,we can write

Ẋ=X-(Components used in the 1st round)

- **Input Data Deflation:Finding the components used in the 1st round**

☐ The input data matrix can be written as a collection of row vector,

$$\begin{Bmatrix} \boldsymbol{x^{(1)}} & \boldsymbol{\dots} & \boldsymbol{x^{(N)}} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{x_1^{(1)}} & \boldsymbol{\dots} & \boldsymbol{x_1^{(N)}} \\ \boldsymbol{\vdots} & \boldsymbol{\ddots} & \boldsymbol{\vdots} \\ \boldsymbol{x_m^{(1)}} & \boldsymbol{\dots} & \boldsymbol{x_m^{(N)}} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{\zeta_1} \\ \boldsymbol{\vdots} \\ \boldsymbol{\zeta_m} \end{Bmatrix}$$

☐ The score of each data point from $x^(1)$ to $x^{(N)}$ can be collectectively represented as

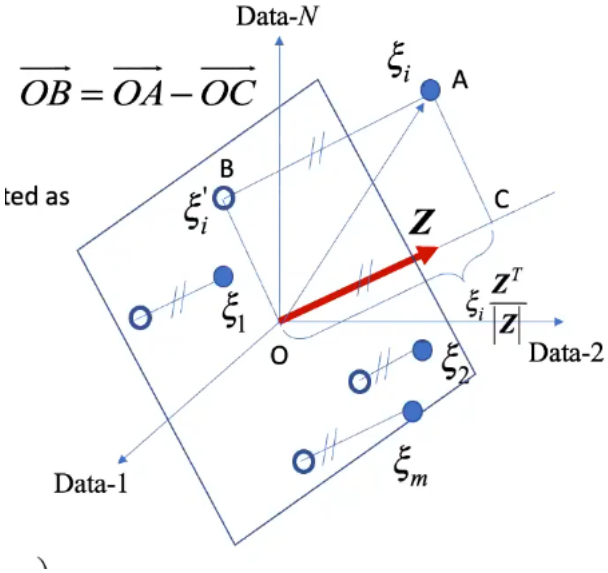
$$\boldsymbol{z = v^T x \longrightarrow \begin{Bmatrix} z^{(1)} & \dots & z^{(N)} \end{Bmatrix} = Z}$$

☐ Plot **Z** in N-dimensional space together with $\zeta_1, \dots, \zeta_m$
☐ The direction of vector **Z** indicates the distribution of scores among the N data points at which the first round latent variables have extracted information from the original data for predicting the output.
☐ We need to delete these components already used in the first round.
☐ Projecting ζ_i onto the plane perpendicular to vector **Z** yields.

$$\boldsymbol{\acute{\zeta}_i = \zeta_i - \zeta_i \frac{Z^T}{|Z|} \times \frac{Z}{|Z|} = \zeta_i \left(I - \frac{Z^T Z}{|Z|^2} \right)}$$

Or collectively, $\acute{X} = X\left(I - \frac{Z^T Z}{|Z|^2}\right)$

Note:
_That vectors Z and ζ_i are row vectors.
_$\zeta_i \frac{Z^T}{|Z|}$:Magnitude , $\frac{Z}{|Z|}$:Direction.



- **Continuing the derivation**
  ☐ Let us rewrite $\acute{X} = X\left(I - \frac{Z^T Z}{|Z|^2}\right.$
  ☐ Note that $\mathbf{Z} = \begin{Bmatrix} z^{(1)} & \dots & z^{(N)} \end{Bmatrix} = \begin{Bmatrix} v^T x^{(1)} & \dots & v^T x^{(N)} \end{Bmatrix} = v^T X$
  ☐ Therefore $\mathbf{|Z|^2 = ZZ^T = v^T X X^T v \cong v^T C_{xx} v}$
  The above deflated data matrix can be written as

$$\dot{\boldsymbol{X}} = \boldsymbol{X}\left(\boldsymbol{I} - \frac{\boldsymbol{Z}^T\boldsymbol{Z}}{|\boldsymbol{Z}|^2}\right) = \boldsymbol{X} - \frac{1}{\boldsymbol{v}^T\boldsymbol{C}_{xx}\boldsymbol{v}}\boldsymbol{X}\boldsymbol{Z}^T\boldsymbol{Z} = \boldsymbol{X} - \frac{1}{\boldsymbol{v}^T\boldsymbol{C}_{xx}\boldsymbol{v}}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{v}^T\boldsymbol{Z}$$

$$= \boldsymbol{X} - \frac{1}{\boldsymbol{v}^T\boldsymbol{C}_{xx}\boldsymbol{v}}\boldsymbol{v}\boldsymbol{v}^T\boldsymbol{X} = \left(\boldsymbol{I} - \frac{\boldsymbol{C}_{XX}\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{C}_{XX}\boldsymbol{v}}\right)\boldsymbol{X}$$

  ☐ For each column vector $\dot{\mathbf{X}} = \left(\mathbf{I} - \frac{\mathbf{C_{XX}vv^T}}{\mathbf{v^T C_{xx} v}}\right)\mathbf{X} = \mathbf{X} - \frac{\mathbf{C_{XX}v}}{\mathbf{v^T C_{XX} v}}\mathbf{v^T X} = \mathbf{X} - \mathbf{pz}$
  Where $\mathbf{p} = \frac{\mathbf{C_{xx}v}}{\mathbf{v^T C_{xx} v}}$ **is called Input loading Vector.**
- **Properties of the Deflated Data Matrix and the Input Loading Vector**
  We can show that, with the input loading vector p,the deflated data matrix becomes the smallest.

$$\mathbf{p^o} = \mathbf{arg}\min_{\mathbf{p}} \mathbf{E}|[\dot{\mathbf{x}}]|^2 \longleftarrow \dot{\mathbf{x}} = \mathbf{x} - \mathbf{zp} = \mathbf{x} - \mathbf{pv^T x} = (\mathbf{I} - \mathbf{pv^T})\mathbf{x}$$

$$= \boldsymbol{arg}\min_{\boldsymbol{p}} \boldsymbol{E}[\boldsymbol{x}^T(\boldsymbol{I} - \boldsymbol{p}\boldsymbol{v}^T)^T(\boldsymbol{I} - \boldsymbol{p}\boldsymbol{v}^T)\boldsymbol{x}]$$

$$= \boldsymbol{arg}\min_{\boldsymbol{p}} \left\{\boldsymbol{E}[\boldsymbol{x}^T\boldsymbol{x}] - 2\boldsymbol{p}^T\boldsymbol{E}[\boldsymbol{x}\boldsymbol{x}^T]\boldsymbol{v} + \boldsymbol{p}^T\boldsymbol{p}\boldsymbol{v}^T\boldsymbol{E}[\boldsymbol{x}\boldsymbol{x}^T]\boldsymbol{v}\right\}$$

  Necessary conditions for min.

$$\frac{\mathbf{d}}{\mathbf{d}\boldsymbol{p}} = \mathbf{0} \quad 2\boldsymbol{C}_{\boldsymbol{XX}}\boldsymbol{v} + 2\boldsymbol{p}\boldsymbol{v}^T\boldsymbol{C}_{\boldsymbol{XX}}\boldsymbol{v} = \mathbf{0}$$

  $\therefore \mathbf{p^o} = \frac{\mathbf{C_{XX}v}}{\mathbf{v^T C_{XX} v}}$
  This is the same as the input loading vector.Therefore,the loading vector minimizes the deflated data matrix.In other words, the 1^{st} round latent variables have taken the most infromation.

_Input Deflation $\hat{\mathbf{x}} = (\mathbf{I} - \mathbf{p^o v^T})\mathbf{x} = \left(\mathbf{I} - \frac{\mathbf{C_{XX} v v^T}}{\mathbf{v^T C_{XX} v}}\right)$ **Output Defation:** $\hat{y} = y - zq^o$

Question: Why is p not aligned with v?

$\Rightarrow$ $If C_{XX}$ is the identity matrix, v and p are aligned.However, the data are distributed not uniformly over the input space:

$$\mathbf{C_{XX}} \neq \mathbf{kI}$$

_Deflated Covariance and Cross-Covariance Matrices: $\mathbf{C_{XX}}$ and $\mathbf{C_{YX}}$

_To compute the second set of latent variables,we need

$$\grave{\boldsymbol{C}}_{\boldsymbol{XX}} = \boldsymbol{E}[\hat{\boldsymbol{x}}(\hat{\boldsymbol{x}})^{\boldsymbol{T}}] \;\; \grave{\boldsymbol{C}}_{\boldsymbol{XX}} = \boldsymbol{E}[\hat{\boldsymbol{y}}(\hat{\boldsymbol{x}})^{\boldsymbol{T}}]$$

_For the deflated input and output data.

$$\grave{\boldsymbol{C}}_{\boldsymbol{XX}} = \boldsymbol{E}[(\boldsymbol{I} - \boldsymbol{pv}^{\boldsymbol{T}})\boldsymbol{xx}^{\boldsymbol{T}}(\boldsymbol{I} - \boldsymbol{vp}^{\boldsymbol{T}})] \;\leftarrow\; \hat{\boldsymbol{x}} = (\boldsymbol{I} - \boldsymbol{pv}^{\boldsymbol{T}})\boldsymbol{x}$$

$$= (\boldsymbol{I} - \boldsymbol{pv}^{\boldsymbol{T}})\boldsymbol{E}[\boldsymbol{xx}^{\boldsymbol{T}}](\boldsymbol{I} - \boldsymbol{vp}^{\boldsymbol{T}})$$

$$= \boldsymbol{C_{XX}} - \boldsymbol{pv}^{\boldsymbol{T}}\boldsymbol{C_{XX}} - \boldsymbol{C_{XX}}\boldsymbol{vp}^{\boldsymbol{T}} + \boldsymbol{pv}^{\boldsymbol{T}}\boldsymbol{C_{XX}}\boldsymbol{vp}^{\boldsymbol{T}}$$

$$= \boldsymbol{C_{XX}} - \boldsymbol{pv}^{\boldsymbol{T}}\boldsymbol{C_{XX}} \;\leftarrow\; \boldsymbol{pv}^{\boldsymbol{T}}\boldsymbol{C_{XX}}\boldsymbol{v} = \boldsymbol{C_{XX}}\boldsymbol{v}, \boldsymbol{p} = \frac{\boldsymbol{C_{XX}}\boldsymbol{v}}{\boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{C_{XX}}\boldsymbol{v}}$$

$$\grave{\boldsymbol{C}}_{\boldsymbol{XX}} = (\mathbf{I} - \mathbf{pv^T})\mathbf{C_{XX}} \;\; Similarly, C'_{YX} = CYX(I - vp^T)$$

- **Partial Least Squares Regression:Summary**

_The most significant m^* sets of latent variable are obtained recursively,

$$\boldsymbol{C_{XX}}[\mathbf{1}] = \boldsymbol{E}[\boldsymbol{xx}^{\boldsymbol{T}}], \;\; \boldsymbol{C_{YX}}[\mathbf{1}] = \boldsymbol{E}[\boldsymbol{yx}^{\boldsymbol{T}}]$$

_For k=1 to m^*

$$\mathbf{C_{XX}}[\mathbf{k+1}] = (\mathbf{I} - \mathbf{p}[\mathbf{k}]\mathbf{v^T}[\mathbf{k}])\mathbf{C_{XX}}[\mathbf{k}];$$

$$\boldsymbol{C_{YX}}[\boldsymbol{k+1}] = \boldsymbol{C_{YX}}[\boldsymbol{k}](\boldsymbol{I} - \boldsymbol{v}[\boldsymbol{k}]\boldsymbol{p}^{\boldsymbol{T}}[\boldsymbol{k}]);$$

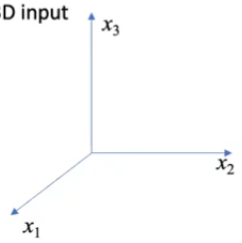
$$\boldsymbol{p}[\boldsymbol{k}] = \frac{\boldsymbol{C_{XX}}[\boldsymbol{k}]\boldsymbol{v}[\boldsymbol{k}]}{\boldsymbol{v}[\boldsymbol{k}]^{\boldsymbol{T}}\boldsymbol{C_{XX}}[\boldsymbol{k}]\boldsymbol{v}[\boldsymbol{k}]} \;\; \boldsymbol{q}[\boldsymbol{k}] = \frac{\boldsymbol{C_{YX}}[\boldsymbol{k}]\boldsymbol{v}[\boldsymbol{k}]}{\boldsymbol{v}[\boldsymbol{k}]^{\boldsymbol{T}}\boldsymbol{C_{XX}}[\boldsymbol{k}]\boldsymbol{v}[\boldsymbol{k}]}$$

_with these loading vectors, x and y can be approximated to

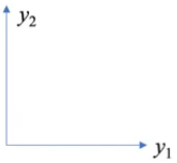
$$\boldsymbol{x} = \sum_{\boldsymbol{k=1}}^{\boldsymbol{m^*}} \boldsymbol{z}[\boldsymbol{k}]\boldsymbol{p}[\boldsymbol{k}] + \boldsymbol{x}[\boldsymbol{m^*}] \;\; \boldsymbol{y} = \sum_{\boldsymbol{k=1}}^{\boldsymbol{m^*}} \boldsymbol{z}[\boldsymbol{k}]\boldsymbol{q}[\boldsymbol{k}] + \boldsymbol{y}[\boldsymbol{m^*}]$$

Example

3D input

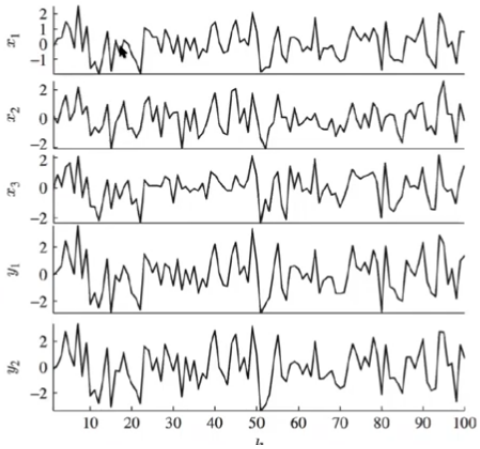


2D output



Sample Data (sample size = 100)

Data are generated with the linear model
+ Gaussian noise $g \sim N(0, 0.05I)$



True relationship

$$y = Bx + g \qquad B = \begin{pmatrix} 0.341 & 0.534 & 0.727 \\ 0.309 & 0.836 & 0.568 \end{pmatrix}$$

Input Covariance

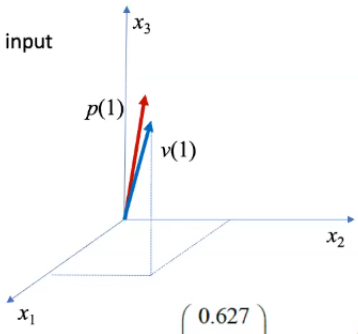
$$C_{XX} = \begin{pmatrix} 1 & 0.8 & 0.9 \\ 0.8 & 1 & 0.5 \\ 0.9 & 0.5 & 1 \end{pmatrix}$$

Input-Output Cross-Covariance

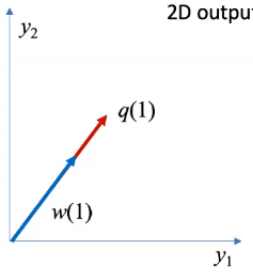
$$C_{YX} = \begin{pmatrix} 1.42 & 1.17 & 1.30 \\ 1.49 & 1.37 & 1.27 \end{pmatrix}$$

First Round Latent Variables

3D input



2D output



$$v(1) = \begin{pmatrix} 0.627 \\ 0.5515 \\ 0.550 \end{pmatrix}, \quad w(1) = \begin{pmatrix} 0.679 \\ 0.734 \end{pmatrix}, \quad p(1) = \begin{pmatrix} 0.631 \\ 0.536 \\ 0.561 \end{pmatrix}, \quad q(1) = \begin{pmatrix} 0.915 \\ 0.989 \end{pmatrix}$$

Singular Value Decomposition of the cross-correlation matrix

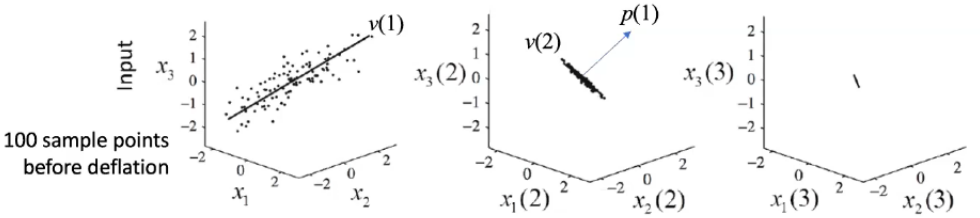
$$C_{XY} = \begin{bmatrix} v_1 \cdots \end{bmatrix} \begin{bmatrix} s_1 & \cdots & 0 \\ \vdots & * & 0 \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} w_1^T \\ \vdots \\ \vdots \end{bmatrix}$$

$$p = \frac{C_{XX} v}{v^T C_{XX} v} \qquad q = \frac{C_{YX} v}{v^T C_{XX} v}$$

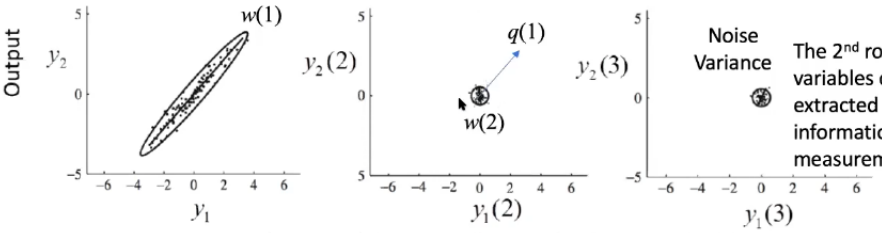
25

Partial Least Squares Regression

1-st round 2-nd round 3-rd round



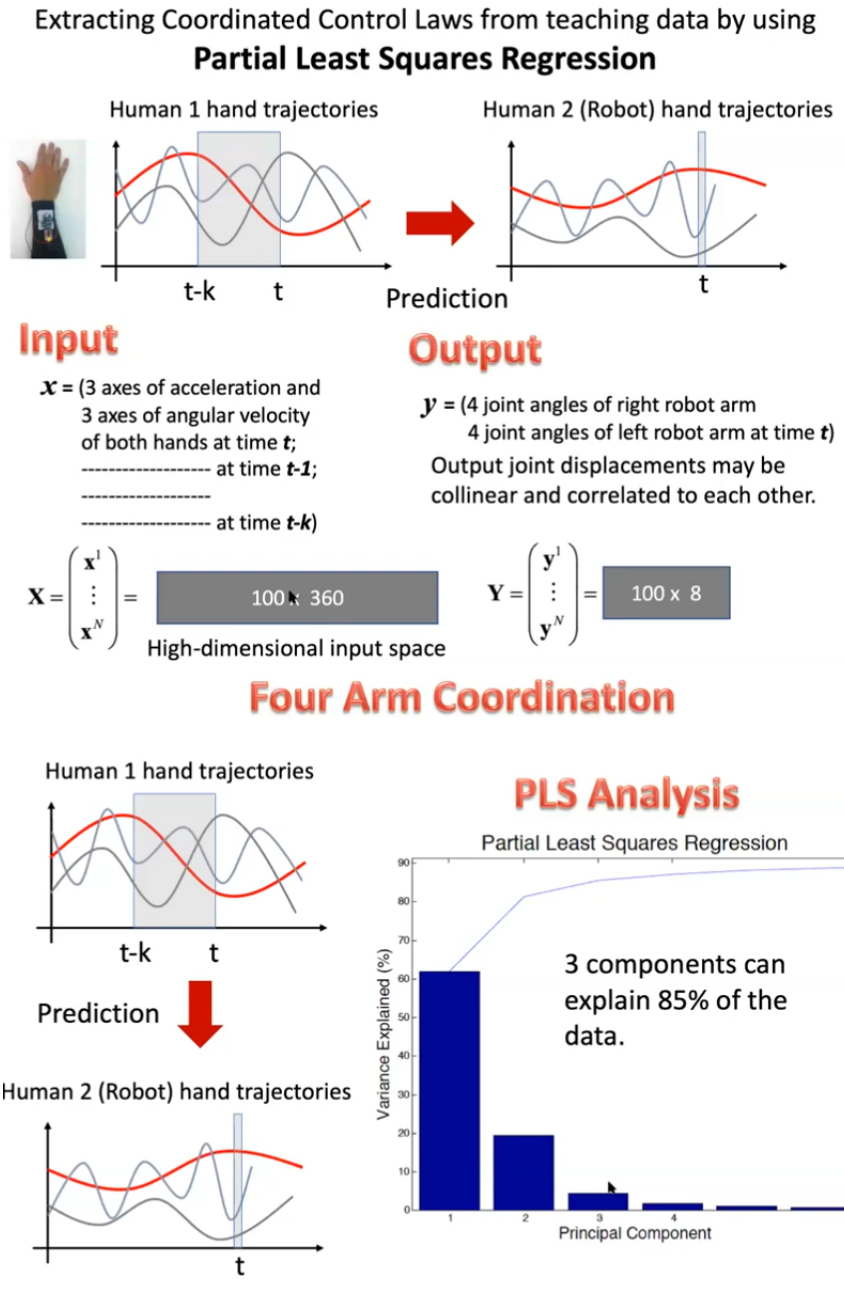
100 sample points before deflation



The 2nd round latent variables completely extracted all meaningful information, leaving only measurement noise.

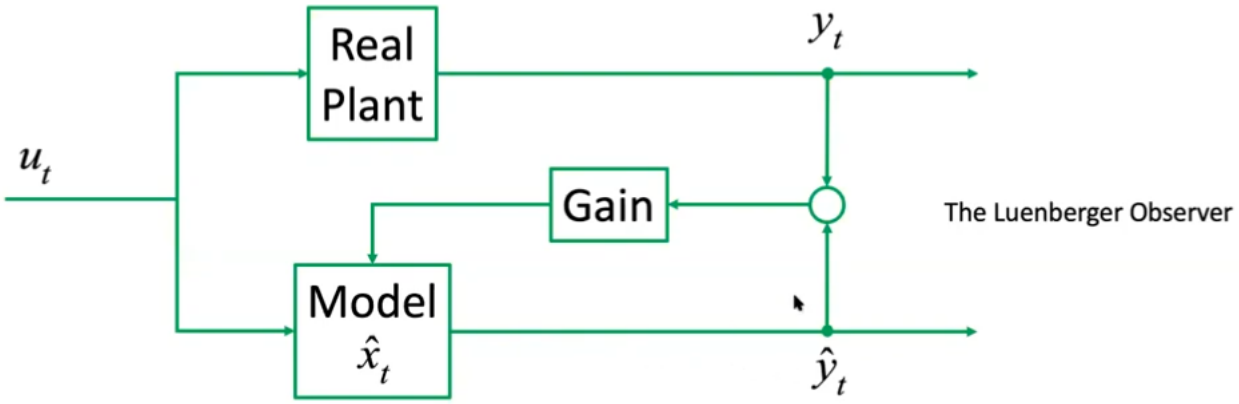
The original data are deflated in the direction of $p(1)$ for input and that of $q(1)$ for output.

26



7. Kalman Filter Discrete time

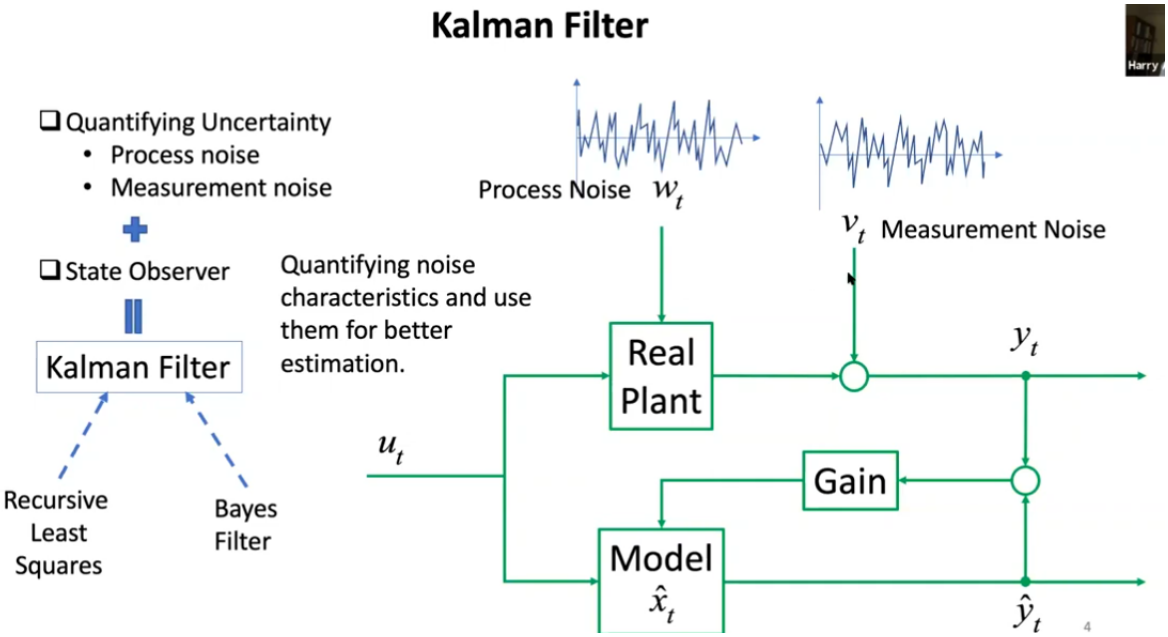
7.1 Prediction Error Correction Formalism



- Estimating state variables, as opposed to plant parameters:State Observer.

$$\hat{\theta} \Rightarrow \hat{x}_t$$

- Real-time recursive computation
  - Prediction error feedback to state estimation



7.2 Discrete-Time State Observer Formulation

- Plant Model:Linear Time-Varying System
  - State (transition) equation

$$\mathbf{X_{t+1}} = \mathbf{A_t x_t} + \mathbf{B_t u_t}$$

where $x_t \in \mathrm{Re}^m \times 1$ State vector $u_t \in \mathrm{Re}^{r\times 1}$ input

  - Output equation (measurement equation)

$$\mathbf{y_t} = \mathbf{H_t x_t} \quad y_t \in \mathrm{Re}_{l\times 1}$$

- Luenberger Observer

$$\mathbf{\hat{x}_{t+1}} = \mathbf{A_t \hat{x}_t} + \mathbf{B_t u_t} + \mathbf{L}(\mathbf{y_t} - \mathbf{\hat{y}_t})$$

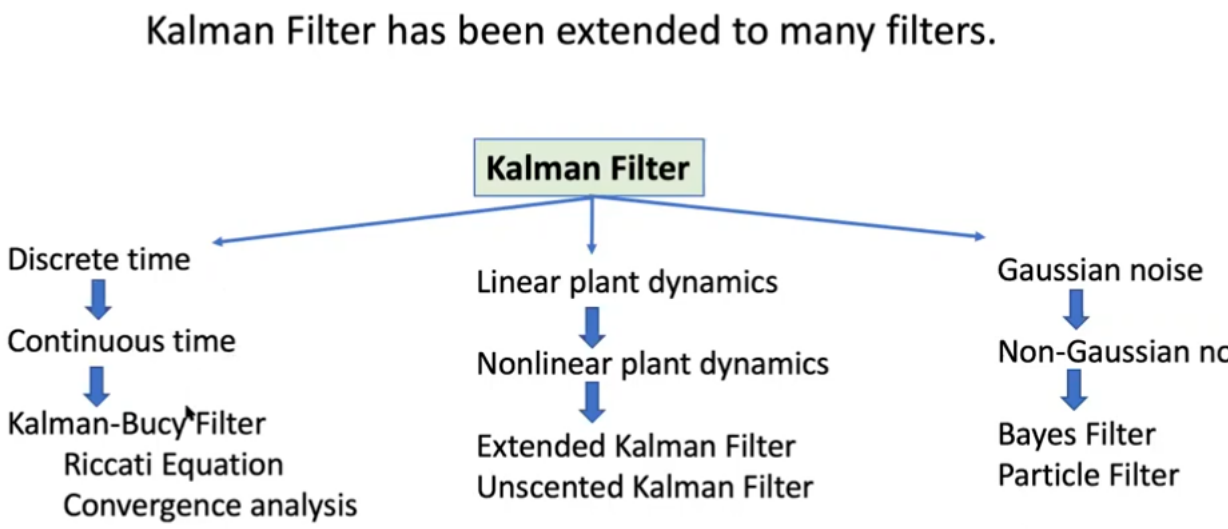
$y_t - \hat{y}_t) \Rightarrow$ Prediction Error: negative feedback

- If the system is observable, the estimated state exponentially converges to the true state.

$$\hat{x}_t \quad \Rightarrow_{t->\infty} \quad x_t$$

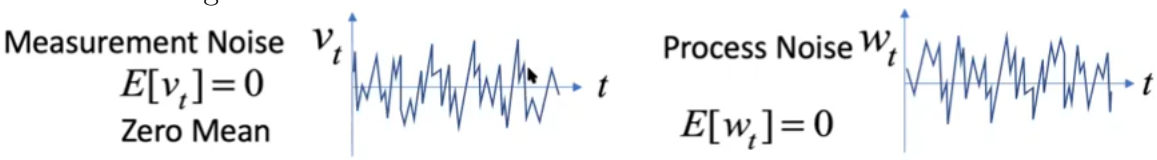
Convergence speed: Pole placement

- **Kalman filter uses an optimal gain based on statival properties of noise.**



7.3 Formulation of Discrete Kalman Filter: Quantification of Uncertainty

- First we quantify both measurement noise and process noise with respect to mean and correlation/covariance.We assume that noise is wide-sense stationary.
- The mean of noise is assumed zero,and constant.If the mean is not zero,then we can shift the origin of coordinate axes.

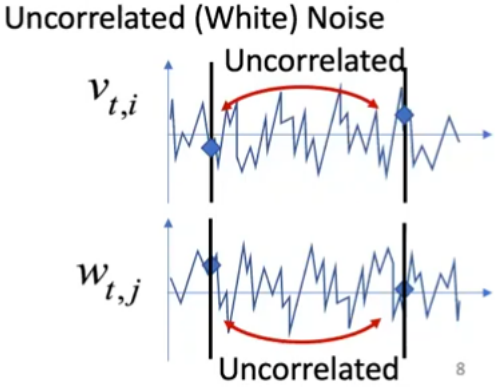


- We assume Uncorrelated (White) noise.
- Note that each of measurement and process noise is a vectorial quantity.Auto-correlation is the correlation between two time slices of the same random process(the same coponent of a noise vector).
  Auto-Correlation

$$\mathbf{E}[\mathbf{v_{t,i}v_{s,i}}] = \mathbf{0}$$

  For all t and s, t≠ s

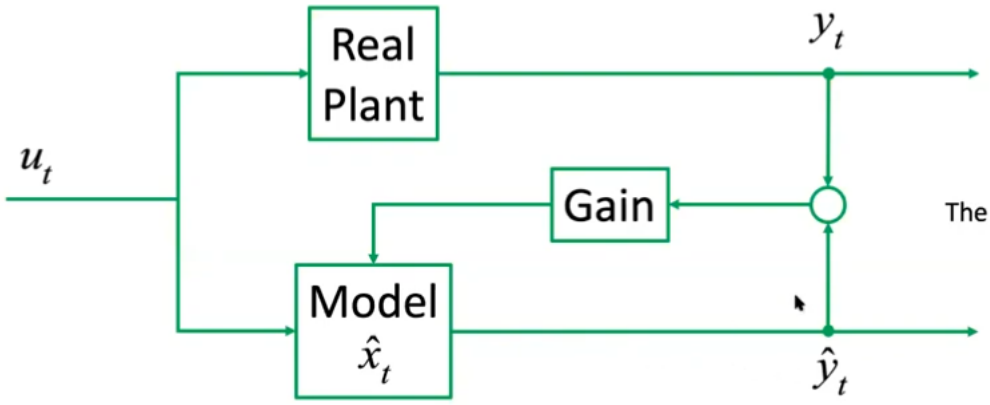
$$\mathbf{E}[\mathbf{w_{t,j}w_{s,j}}] = \mathbf{0}$$



7.4 Quantification of Uncertainty:Cross-Correlation

- We also characterize noise properties with respect to Cross-Correlation:the correlation between two different components of the same noise vector or the one

between measurement and process noise.



7.5 Quantification of Uncertainty:Covariance Matrices

- In vector and matrix form, auto-correlation and cross-correlation can be collectively expressed as

$$\boldsymbol{E}[\boldsymbol{v_t v_s^T}] = \begin{Bmatrix} \boldsymbol{E}[\boldsymbol{v_{t,1}v_{s,1}}] & \dots & \boldsymbol{E}[\boldsymbol{v_{t,1}v_{s,l}}] \\ \vdots & \ddots & \vdots \\ \boldsymbol{E}[\boldsymbol{v_{t,l}v_{s,1}} & \dots & \boldsymbol{E}[\boldsymbol{v_{t,l}v_{s,l}}]] \end{Bmatrix}$$

- When t=s, the diagonal terms represent variance of the individual noise component and off-diagonal terms co-variance. $\mathbf{E}[\mathbf{v_{t,i}^2}] = \sigma_\mathbf{i}^\mathbf{2}$
- In summary,
  - Measurement noise covariance

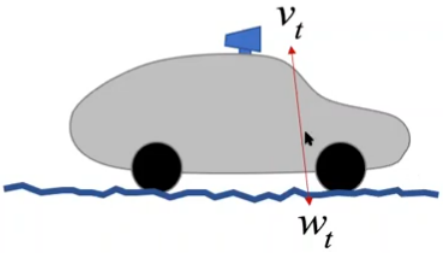
$$\mathbf{E}[\mathbf{v_t v_s^T}] = \begin{cases} R_t; & t = s \\ 0; & t \neq s \end{cases}$$

  - Covariance R_t is assumed to be positive definite.
  - There is no perfect sensor.
  - Process noise covariance

$$\mathbf{E}[\mathbf{w_t w_s^T}] = \begin{cases} Q_t; & t = s \\ 0; & t \neq s \end{cases}$$

  - Covariance Q_t is assumed to be positive semi-definite.

Measurement and Process Noise Cross-Correlation



PS#2 Problem 2

$E[w_t v_s^T] = 0$ For all t and s,

- However, in some application, process noise also influences measurement, as in the case of a self-driving car.
- For the sake of simplicity, we assume that there is no correlation between them, but this assumption can be removed.

- Finally,we assume that these noise terms additively disturb the process.Namely the **state and measurement equations** are given by

$$\mathbf{x_{t+1}} = \mathbf{A_t x_t} + \mathbf{B_t u_t} + \mathbf{G_t w_t}$$

$$\mathbf{y_t} = \mathbf{H_t x_t} + \mathbf{v_t}$$

$\mathbf{v_t}, \mathbf{G_t w_t}$=Additive noise

- G_t represents how the process noise disturb the state variables

7.6 Optimal Filtering Problem

- Find a state estimate, $\hat{X}_t$, that minimizes the mean squared prediction error:

$$\bar{\mathbf{J}}_\mathbf{t} = \mathbf{E}[|\hat{\mathbf{x}}_\mathbf{t} - \mathbf{x}_\mathbf{t}|^\mathbf{2}]$$

Subject to state and measurement equations

$$x_{t+1} = A_t x_t + B_t u_t + G_t w_t$$

. Linear Time Varying System

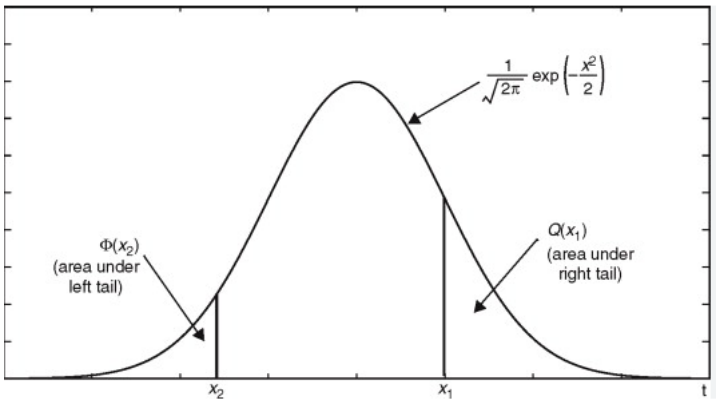
$$y_t = H_t x_t + v_t$$

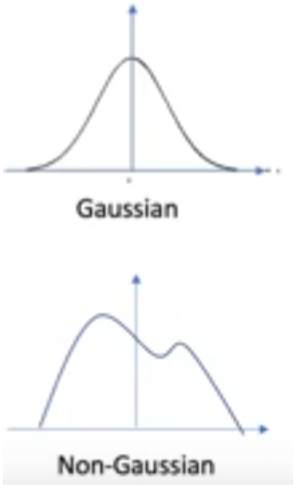
Where process noise, w_t, and measurement noise, v_t, are uncorrelated(White)noise as characterized above.

- Assuming that the noise distribution is Gaussian, Kalman Filter is the optimal among linear and nonlinear filter.

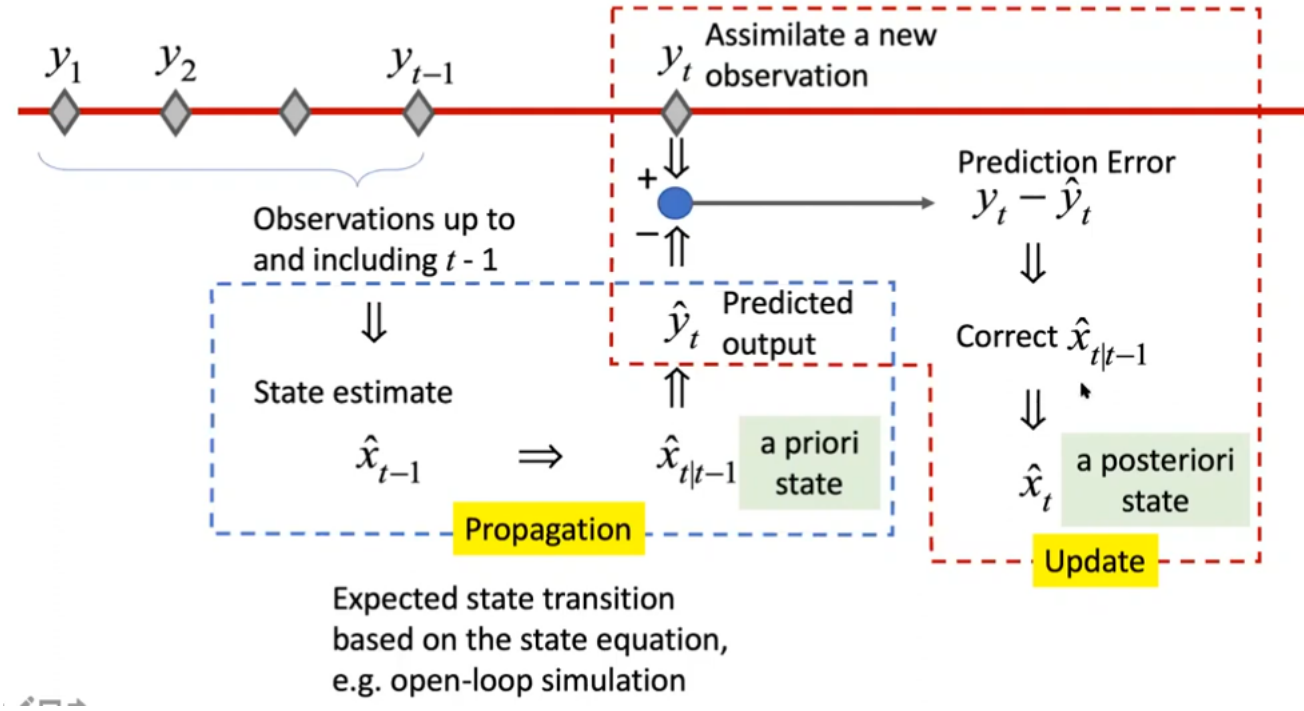
$$\hat{\mathbf{x}}_{\mathbf{t+1}} = \mathbf{A}_\mathbf{t}\hat{\mathbf{x}}_\mathbf{t} + \mathbf{B}_\mathbf{t}\mathbf{u}_\mathbf{t} + \mathbf{K}_\mathbf{t}[\mathbf{y}_\mathbf{t} - \hat{\mathbf{y}}_\mathbf{t}]$$

- Assuming that the filter structure is linear, $K_t[y_t - \hat{y}_t]$ Kalman Filter is the optimal linear filter,regardless of noise distribution.
We first prove the second problem, and show the proof for the first problem, and show the proof for the first problem at the Bayes Filter lecture.





7.7 The flow of the Discrete Kalman Filter Algorithm



7.8 Propagation of State

- Using the state equation, we want to predict the transition of state:
- The deterministic term, $B_{t-1}u_{t-1}$,can be omitted by setting, $\mathbf{u_{t-1}} = 0$,Without loss of generality
- Taking expectation yields

$$\mathbf{E[x_t] = E[A_{t-1}x_{t-1} + G_{t-1}] = A_{t-1}E[x_{t-1}] + G_{t-1}E[w_{t-1}]}$$

- This expected value of state,$E[x_t]$,is the predicted state at time t based on the estimated state at t-1.This is **a priori** state estimate denoted by

$$\mathbf{\hat{x}_{t|t-1} = A_{t-1}\hat{x}_{t-1}}$$

- Using the output equation, the expected output is constructed as

$$\mathbf{\hat{y}_t = E[y_t] = E[H_t x_t + v_t] = H_t E[x_t] + E[v_t]}$$

$$\mathbf{\therefore \hat{y}_t = H_t\hat{x}_{t|t-1} = H_t A_t \hat{x}_{t-1}}$$

- This is a predicted output to be compared to an actual measured output.

7.9 State Update

- Let y_t be a newly observed output.The predicted output $\hat{y}_t$ based on the previous state estimate is then compared to the actual measured output and the error is used for correcting,or updating,the a priori state estimate:

$$\mathbf{\hat{x}_t = \hat{x}_{t|t-1} + K_t[y_t - \hat{y}_t]} \quad \mathbf{(23)}$$

- Note that we assume this linear update law in this proof. Gain $K_t \in \mathrm{Re}^m$ is called the Kalman gain. Our goal is to find an optimal gain that minimizes the mean squared prediction error: $\bar{J}_t = E[|\hat{x}_t - x_t|^2]$
- Optimal gain:

$$\mathbf{K_t = \arg\min_{K_t} \bar{J}_t = \arg\min_{k_t} E[|\hat{x}_t - x_t|^2]}$$

7.10 Effect of Kalman Gain K_t

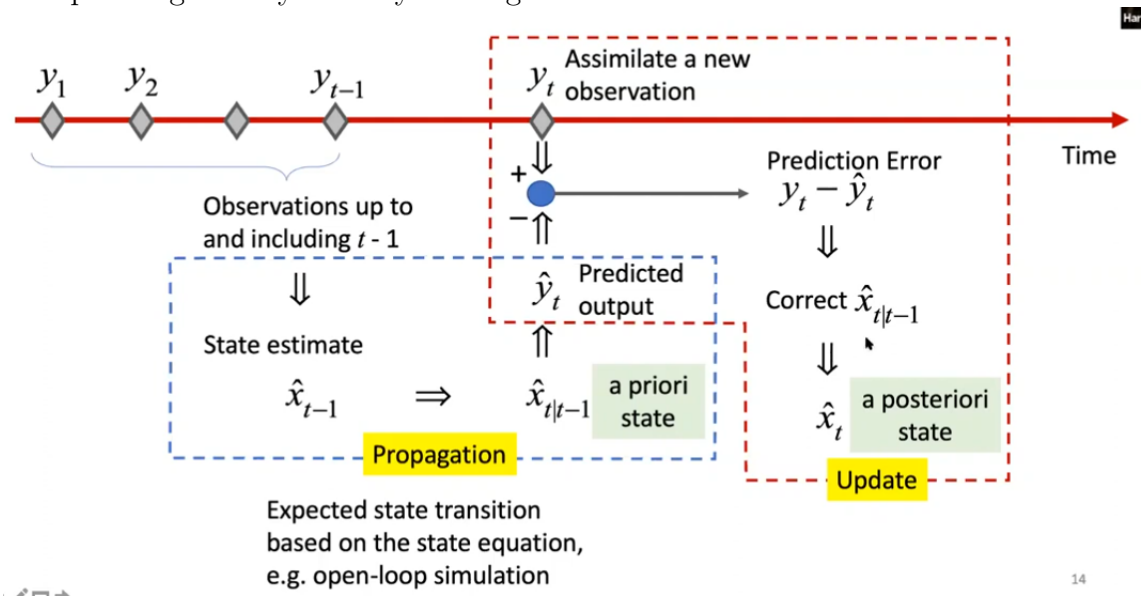
- Prediction Law: $\hat{\mathbf{x}}_\mathbf{t} = \hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}} + \mathbf{K_t}[\mathbf{y_t} - \hat{\mathbf{y}}_\mathbf{t}]$ (1)
- Note that there are two types of prediction error.
  - A posteriori error: $\boldsymbol{e_t} \triangleq \hat{\boldsymbol{x}}_\boldsymbol{t} - \boldsymbol{x_t}$
  - A priori error: $\boldsymbol{\varepsilon} \triangleq \hat{\boldsymbol{x}}_{\boldsymbol{t|t-1}} - \boldsymbol{x_t}$
- These two prediction errors are related to each.Using the state update law (1),

$$\mathbf{e_t} = \hat{\mathbf{x}}_\mathbf{t} - \mathbf{x_t} = \hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}} + \mathbf{K_t}[\mathbf{y_t} - \hat{\mathbf{y}}_\mathbf{t}] - \mathbf{x_t}$$

$$= \hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}} + \mathbf{K_t}[\mathbf{H_t x_t} + \mathbf{v_t} - \mathbf{H_t}\hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}}] - \mathbf{x_t}$$

$$= \hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}} - \mathbf{x_t K_t H_t}(\mathbf{x_t} - \hat{\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}}) + \mathbf{K_t v} - \mathbf{t}$$

- As the gain K_t becomes higher,the a prior error is more reduced but the measurement noise is amplified.
- An optimal gain may exist by making the trade-off between the two



7.11 Computation of an optimal gain

- Computation of the squared a posteriori error: $\bar{\mathbf{J}}_\mathbf{t} = \mathbf{E}[|\hat{\mathbf{x}}_\mathbf{t} - \mathbf{x_t}|^\mathbf{2}]$
- Recall $\mathbf{e_t} = \hat{\mathbf{x}}_\mathbf{t} - \mathbf{x_t} = (\mathbf{I} - \mathbf{K_t H_t})\varepsilon_\mathbf{t} + \mathbf{K_t v_t}$
- Omitting t for brevity

$$|\boldsymbol{e}|^\mathbf{2} = [(\boldsymbol{I} - \boldsymbol{KH})\boldsymbol{\varepsilon} + \boldsymbol{Kv}]^T[(\boldsymbol{I} - \boldsymbol{KH})\boldsymbol{\varepsilon} + \boldsymbol{Kv}]$$

$$= \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} + \boldsymbol{\varepsilon}^T\boldsymbol{H}^T\boldsymbol{K}^T\boldsymbol{KH\varepsilon} + \boldsymbol{v}^T\boldsymbol{K}^T\boldsymbol{Kv} - \mathbf{2}\boldsymbol{\varepsilon}^T\boldsymbol{KH\varepsilon} - \mathbf{2}\boldsymbol{\varepsilon}^T\boldsymbol{H}^T\boldsymbol{K}^T\boldsymbol{Kv} + \mathbf{2}\boldsymbol{\varepsilon}^T\boldsymbol{Kv}$$

- Necessary conditions for min $\bar{\mathbf{J}}_\mathbf{t}(\mathbf{K})$

$$\frac{\mathrm{d}\bar{\mathbf{J}}_\mathbf{t}}{\mathrm{d}\mathbf{K}} = \mathbf{0}$$

But, $K = \{K_{ij}\}$ is a matrix

$$\frac{\mathrm{d}\bar{\mathbf{J}}_\mathbf{t}}{\mathrm{d}\mathbf{K_y}} = \mathbf{0}$$

For all i and j, $\mathbf{1} \leq \mathbf{i} \leq \mathbf{n}, \mathbf{1} \leq \mathbf{j} \leq \mathbf{l}$

7.12 Lemma Matrix differentiation Rules

- Consider a scalar function: $\mathbf{f} = \mathbf{a^T K b} = \sum_i \sum_j \mathbf{a_i b_j K_{ij}}$
  Where $\mathbf{a} \in \mathrm{Re}^{\mathbf{m\times 1}}, \mathbf{b} \in \mathrm{Re}^{\mathbf{l\times 1}}, \mathbf{K} \in \mathrm{Re}^{\mathbf{m\times l}}$

$$\frac{\partial f}{\partial K_{pq} = a_p b_p}$$

  Because all others $i \neq p, j \neq q$ are zeros,when differentiating by K_pq.

$$\therefore \frac{\mathbf{df}}{\mathbf{dK}} = \left\{\frac{\partial \mathbf{f}}{\partial \mathbf{K_{pq}}}\right\} = \mathbf{ab^T} \quad \text{Note:} \begin{Bmatrix} \mathbf{a_1} \\ \vdots \\ \mathbf{a_n} \end{Bmatrix} \begin{Bmatrix} \mathbf{b_1} & \dots & \mathbf{b_l} \end{Bmatrix} = \begin{Bmatrix} \mathbf{a_p} & \mathbf{b_q} \end{Bmatrix}$$

- Consider another scalar function:

$$\mathbf{g} = \mathbf{c^T K^T K b}, \ \ \mathbf{c} \in \mathrm{Re}^{\mathbf{l\times 1}}$$

  Similarly, we can show that $\frac{\mathbf{d}}{\mathbf{dK}} = \boldsymbol{Kbc^T + Kcb^T}$

7.13 Computation of Optimal Gain K_t

$$|\boldsymbol{e}|^{\mathbf{2}} = \boldsymbol{\varepsilon^T \varepsilon + \varepsilon^T H^T K^T K H \varepsilon + v^T K^T K v - 2\varepsilon^T K H \varepsilon - 2\varepsilon^T H^T K^T K^T K v + 2\varepsilon^T K v}$$

- Applying these rules of differentiation by matrix K to the derivative of squared error,

$$\frac{\mathbf{d}|\boldsymbol{e}|^{\mathbf{2}}}{\mathbf{d}\boldsymbol{K}} = \frac{\mathbf{d}}{\mathbf{d}\boldsymbol{K}}\boldsymbol{\varepsilon^T \varepsilon} + \frac{\mathbf{d}}{\mathbf{d}\boldsymbol{K}}\mathbf{2}\boldsymbol{\varepsilon^T K v} - \mathbf{2}\boldsymbol{\varepsilon^T K H \varepsilon} \quad \dots \ \boldsymbol{Rule1}$$

$$+\frac{\mathbf{d}}{\mathbf{d}\boldsymbol{k}}\boldsymbol{\varepsilon^T H^T K^T K H \varepsilon + v^T K^T K v} - \mathbf{2}\boldsymbol{\varepsilon^T H^T K^T K v} \quad \dots \boldsymbol{Rule2}$$

$$= \mathbf{2}^{\boldsymbol{T}} - \mathbf{2}\boldsymbol{\varepsilon\varepsilon^T H^T + K H \varepsilon\varepsilon^T H^T + 2K v v^T - 2K v \varepsilon^T H^T - 2K H^T}$$

Rule1 $\frac{\mathbf{d}}{\mathbf{d}\boldsymbol{K}} = \boldsymbol{ab^T}$
Rule2 $\frac{\mathbf{d}}{\mathbf{d}\boldsymbol{K}} = \boldsymbol{Kbc^T + Kcb^T}$

$$\frac{\mathbf{d}|\boldsymbol{e}|^{\mathbf{2}}}{\mathbf{d}\boldsymbol{K}} = \mathbf{2}\boldsymbol{\varepsilon v^T - 2\varepsilon\varepsilon^T H^T + K H \varepsilon\varepsilon^T H^T + 2K v v^T - 2K v \varepsilon^T H^T - 2K H \varepsilon v^T}$$

- Taking expectation and setting it to zero,

$$E[\varepsilon_t v_t^T] - \underbrace{E[\varepsilon_t \varepsilon_t^T]}_{P_{t|t-1}} H_t^T + K_t H_t E[\varepsilon_t \varepsilon_t^T] H_t^T + K_t \underbrace{E[v_t v_t^T]}_{R_t} - K_t E[v_t \varepsilon_t^T] H_t^T = 0$$

Measurement noise covariance

- Define a priori error covariance: $\boldsymbol{P_{t|t-1}} \triangleq \boldsymbol{E[\varepsilon_t \varepsilon_t^T]}$
- Examine $\boldsymbol{E[\varepsilon_t v_t^T]}$

Examine $E[\varepsilon_t v_t^T]$

$$E[\varepsilon_t v_t^T] = E[(\hat{x}_{t|t-1} - x_t)v_t^T] \quad \leftarrow \varepsilon_t = \hat{x}_{t|t-1} - x_t$$

$$= A_{t-1}E[\hat{x}_{t-1}v_t^T] - E[x_t v_t^T] \quad \leftarrow \hat{x}_{t|t-1} = A_{t-1}\hat{x}_{t-1}$$

Examine

$$x_t = A_{t-1}x_{t-1} + G_{t-1}w_{t-1}$$

$$E[x_t v_t^T] = E[(A_{t-1}x_{t-1} + G_{t-1}w_{t-1})v_t^T]$$

$$= A_{t-1}E[x_{t-1}v_t^T] + G_{t-1}E[w_{t-1}v_t^T] \quad \therefore E[x_t v_t^T] = 0$$

From x_{t-1}, only process noise terms w_{t-2}, w_{t-3}, come out, which do not correlate with the measurement noise v_t. $E[w_t v_s^T] = 0$ For all t and s,

Examine

$$E[\hat{x}_{t-1}v_t^T] = E[(\hat{x}_{t-1|t-2} + K_{t-1}(y_{t-1} - \hat{y}_{t-1}))v_t^T] \quad \leftarrow \hat{x}_{t-1} = \hat{x}_{t-1|t-2} + K_{t-1}(y_{t-1} - \hat{y}_{t-1})$$

$$= E[(A_{t-2}\hat{x}_{t-2} + K_{t-1}(H_{t-1}x_{t-1} + v_{t-1} - \hat{y}_{t-1})v_t^T] \quad \leftarrow y_{t-1} = H_{t-1}x_{t-1} + v_{t-1}$$

$$= A_{t-2}E[\hat{x}_{t-2}v_t^T] + K_{t-1}H_{t-1}E[x_{t-1}v_t^T] + E[v_{t-1}v_t^T] - E[\hat{y}_{t-1}v_t^T] = 0$$

v_t does not correlate with past $\hat{x}_{t-2}, x_{t-1}, \hat{y}_{t-1}$ Uncorrelated $E[v_t v_s^T] = 0; \quad t \neq s$

22

- From the above examination: $\boldsymbol{E[\varepsilon_t v_t^T] = 0}$
- Back to the optimality conditions:

$$E[\varepsilon_t v_t^T] - E[\varepsilon_t \varepsilon_t^T]H_t^T + K_t H_t E[\varepsilon_t \varepsilon_t^T]H_t^T + K_t E[v_t v_t^T] - K_t E[v_t \varepsilon_t^T]H_t^T = 0$$

$E[\varepsilon_t \varepsilon_t^T] \rightarrow P_{t|t-1}$, $E[v_t v_t^T] \rightarrow R_t$

$$-P_{t|t-1}H_t^T + K_t H_t P_{t|t-1}H_t^T + K_t R_t = 0$$

Positive definite

- The optimal gain can be obtained from: $\mathbf{K_t(H_t P_{t|t-1} H_t^T + R_t) = P_{t|t-1} H_t^T}$
- Note that matrix $\mathbf{H_t P_{t|t-1} H_t^T + R_t}$ is positive-definite and invertible:

$$\mathbf{K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1}: \quad ThisistheKalmanGain}$$

7.14 Recursive Fomula of Covariances,$\mathbf{P_{t|t-1}}$ and P_t

- The previous expression can be further simplified by using the optimal(Kalman) gain solution.

$$P_t = (I - K_t H_t)P_{t|t-1}(I - K_t H_t)^T + K_t R_t K_t^T \Rightarrow P_t = (I - K_t H_t)P_{t|t-1}$$

$$K_t = P_{t|t-1}H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1}$$

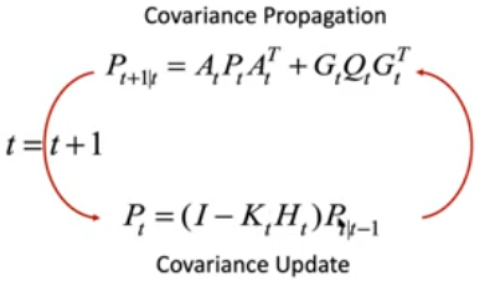
- This implies that a priori covariance is reduced by updating the a priori state estimate $\mathbf{xt|\hat{t} - 1}$ with a newly assimlilated measurement, $\mathbf{y_t}$

This is called Covariance Update:$\mathbf{P_{t|t-1} \Rightarrow P_t}$

- In turn, a priori covariance $P_{t+1|t}$ can be derived from a posteriori covariance P_t
- Recall: $\boldsymbol{\varepsilon_{t+1} = \hat{x}_{t+1|t} - x_{t+1} = A_t\hat{x}_t - (A_t x_t + G_t w_t) = A_t e_t - G_t w_t}$
- Compute: $\mathbf{P_{t+1|t} = E[\varepsilon_{t+1}\varepsilon_{t+1}^T] = E[(A_t e_t - G_t w_t)(A_t e_t - G_t w_t)^T]}$

$$= \mathbf{A_t E[e_t e_t^T]A_t^T + G_t E[w_t w_t^T]G_t^T - A_t E[e_t w_t^T]G_t^T - G_t E[w_t e_t^T]A_t^T}$$

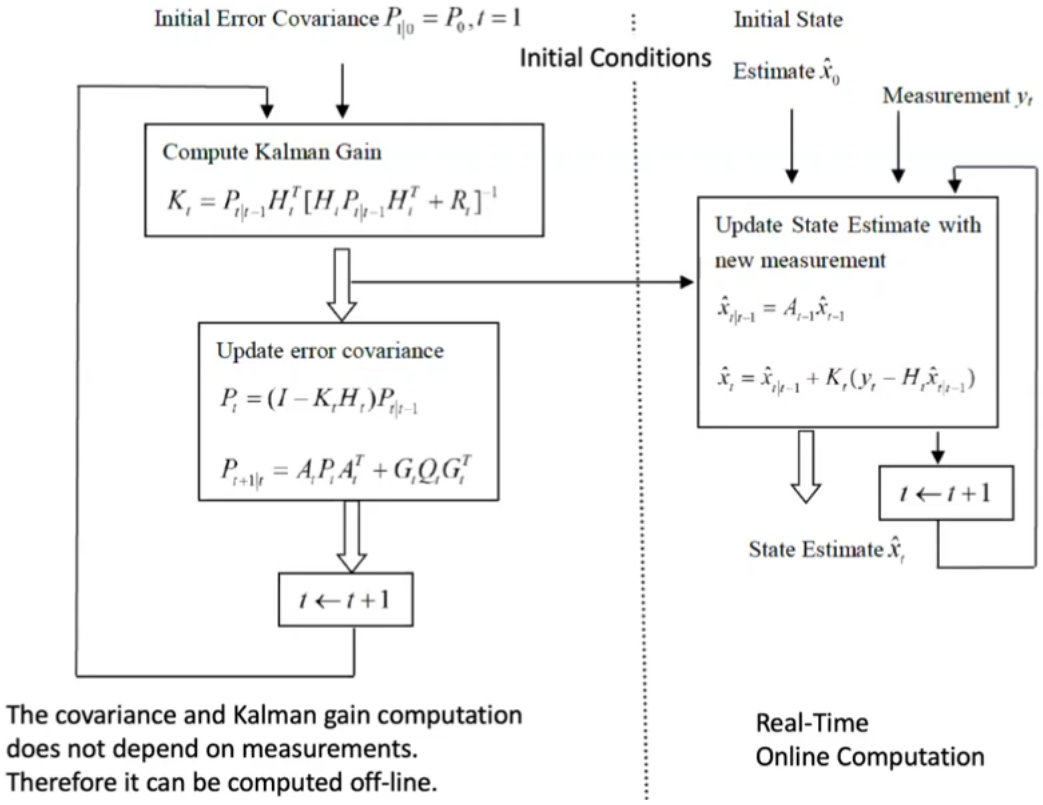
- We can show $\mathbf{E[e_t w_t^T] = 0}$. Therefore, $\mathbf{P_{t+1|t} = A_t P_t A_t^T + G_t Q_t G_t^T}$
- The last formular is called Covariance Propagation: $P_t \Rightarrow P_{t+1|t}$

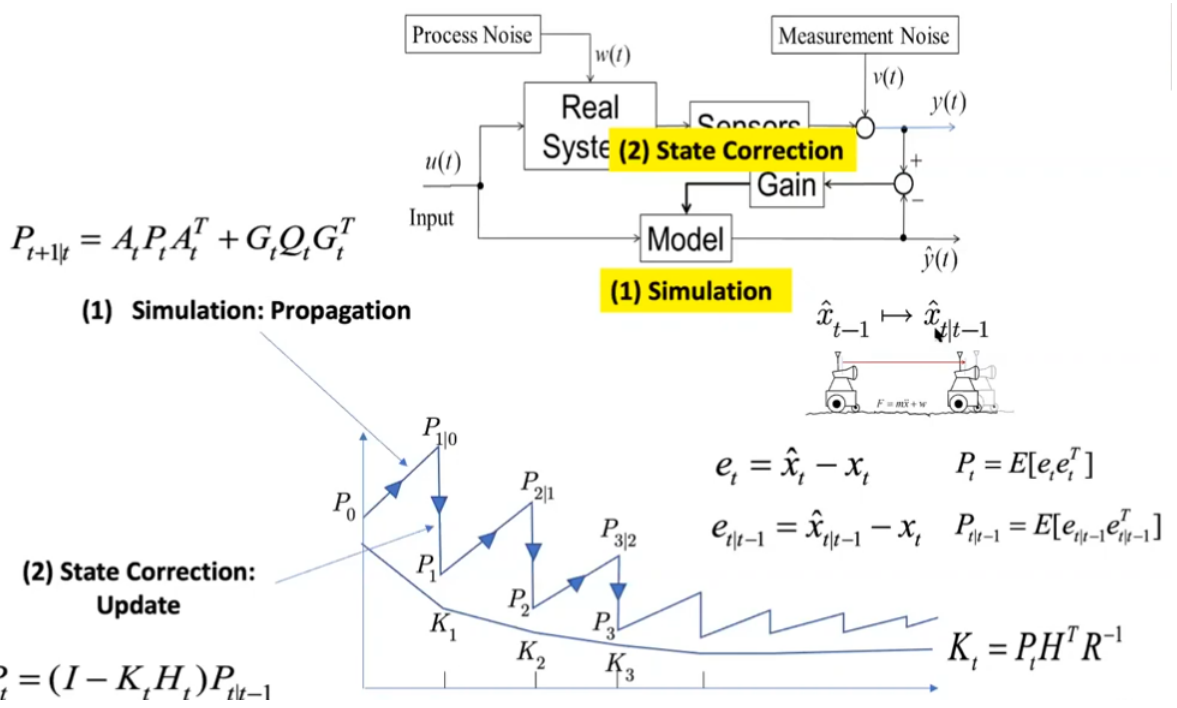


- Given initial coditions, $\mathbf{P_{1|0}}$, covariance matrices can be computed recursively along with the Kalman gain

$$\mathbf{K_t} = \mathbf{P_{t|t-1}H_t^T}(\mathbf{H_tP_{t|t-1}H_t^T} + \mathbf{R_t})^{-\mathbf{1}}$$

7.15 Recursive Computation Algorithm





8. **Continuous Time Kalman Filter**

9. **Linerized Kalman Filter(LKF) ,Extended Kalman Filter(EKF) and Unscented Kalman Filter(UKF)**

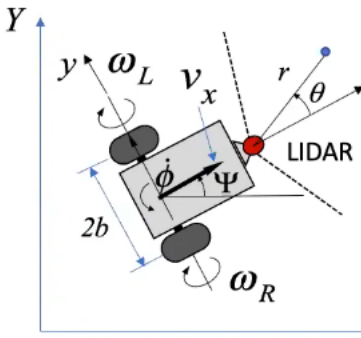
9.1 Applying Kalman Filter to Nonlinear Dynamical Systems

- So far,we hace been dealing with linear dynamical systems for constructing kalman filter.
- However,practical systems are nonlinear to some extent.
  Example
  Vehicle kinematics is nonlinear

$$v_x = \frac{D}{4}(\omega_R + \omega_L) \quad \dot{X} = v_x \cos\Psi$$
$$v_y = 0 \quad \dot{Y} = v_x \sin\Psi$$
$$\dot{\phi} = \frac{D}{4b}(\omega_R - \omega_L) \quad \dot{\Psi} = \dot{\phi}$$

Nonlinear State Equation

$$\dot{x} = f(x,u,t) + w(t)$$



LIDAR (a range finder) attached to a vehicle is in a polar coordinate system: nonlinear.

$$x = r\cos\theta$$
$$y = r\sin\theta$$

Nonlinear Measurement Equation

$$y(t) = h(x,t) + v(t)$$

9.2 Extension of Kalman Filter to Nonlinear Dynamical Systems

- State Equation $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{t}) + \mathbf{w}(\mathbf{t}), \mathbf{x} \in \mathrm{Re}^{\mathbf{m\times 1}}, \mathbf{w} \in \mathrm{Re}^{\mathbf{m\times 1}}$
- Measurement Equation $\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{t}) + \mathbf{v}(\mathbf{t}), \mathbf{y} \in \mathrm{Re}^{\mathbf{l\times 1}}, \mathbf{v} \in \mathrm{Re}^{\mathbf{lx1}}$
- Process noise and measurement noise are uncorrelated,white noise, with variance, Q(t) and R(t).
  We consider three methods
  – Linearized Kalman Filter

- Extended Kalman Filter
- Unscented Kalman Filter

9.3 Linearized Kalman Filter

- Linearize the nonlinear state equation around a nominal trajectory, e.g. reference trajectory, planned trajectory, commanded trajectory.
- The nominal trajectory must satisfy the original nonlinear state equation.

$$\dot{\mathbf{x}}^* = \mathbf{f}(\mathbf{x}^*, \mathbf{u}, \mathbf{t})$$

- Consider deviation from the nominal trajectory:

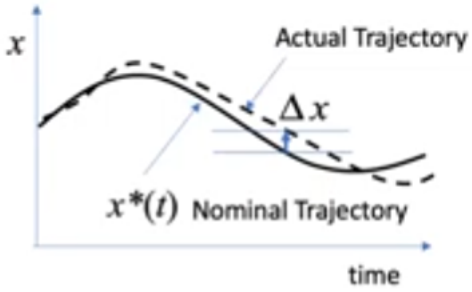
$$\mathbf{x} = \mathbf{x}^* + \boldsymbol{\Delta}\mathbf{x}$$

- Assuming that the deviation from the nominal trajectory is kept small,we can linerize the nonlinear state equation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{t}) + \mathbf{w}(\mathbf{t}) = \mathbf{f}(\mathbf{x}^* + \boldsymbol{\Delta}\mathbf{x}, \mathbf{u}, \mathbf{t}) + \mathbf{w}(\mathbf{t})$$

$$= \boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{t}) + \frac{\boldsymbol{\partial f}}{\boldsymbol{\partial x}}|_{\boldsymbol{x}^*}\boldsymbol{\Delta x} + \boldsymbol{w(t)} + (\boldsymbol{higher - ordersmallquantity})$$

$$\dot{\mathbf{x}}^* = \mathbf{f}(\mathbf{x}^*, \mathbf{u}, \mathbf{t})$$



- Note that the nominal trajectory satisfies the state equation with no process noise and that the derivative of the deviation is given by

$$\dot{x} = f(x^*,u,t) + \left.\frac{\partial f}{\partial x}\right|_{x^*} \Delta x + w(t)$$

$$\dot{x} = \dot{x}^* + \Delta\dot{x} \qquad \dot{x}^* = f(x^*,u,t)$$

$$\Delta\dot{x} \cong \left.\frac{\partial f}{\partial x}\right|_{x^*} \Delta x + w(t)$$

- Now replacing $\triangle\dot{x}$ by and the jacobian matrix $\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial x}}|_{\boldsymbol{x}^*}\boldsymbol{byF(t)}$, we have a linear time-varying state equation

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{t})\mathbf{x}(\mathbf{t}) + \mathbf{w}(\mathbf{t})$$

- Similarly, the measurement equation can be linerized around the nominal trajectory:

$$\boldsymbol{y}^* + \boldsymbol{\Delta y} \cong \boldsymbol{h}(\boldsymbol{x}^*, \boldsymbol{t}) + \frac{\boldsymbol{\partial h}}{\boldsymbol{\partial x}}|_{\boldsymbol{x}^*}\boldsymbol{\Delta x} + \boldsymbol{v(t)}$$

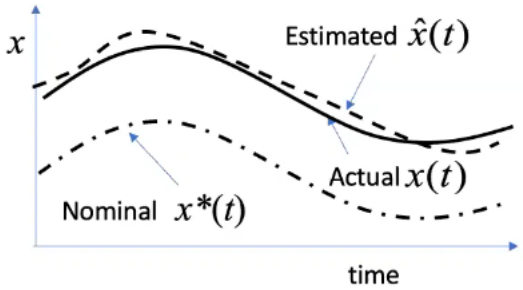
- Note again $\boldsymbol{y^* = h(x^*, t)}$. Replacing $\boldsymbol{\Delta y}$ by y and the Jacobian matrix $\frac{\partial h}{\partial x}|_{x^*}$ by H(t),
- The original Linear Kalman Filter can be applied to this linear time-varying system .
  - State Equation $\dot{\boldsymbol{x}} = \boldsymbol{f(x, u, t)} + \boldsymbol{w(t)}, \Rightarrow \dot{\boldsymbol{x}} = \boldsymbol{F(t)x(t)} + \boldsymbol{w(t)}, \boldsymbol{F(t)} = \frac{\partial f}{\partial x}|_{x^*}$
  - Measurement Equation $\boldsymbol{y = h(x, t)} + \boldsymbol{v(t)}, \Rightarrow \boldsymbol{y(t) = H(t)x(t)} + \boldsymbol{v(t)}, \boldsymbol{H(t)} = \frac{\partial h}{\partial x}|_{x^*}$
  - State propagation and update $\frac{\mathrm{d}}{\mathrm{d}t}\hat{\boldsymbol{x}}(t) = \boldsymbol{F(t)}\hat{\boldsymbol{x}}(t) + \boldsymbol{K(t)}[\boldsymbol{y(t)} - \hat{\boldsymbol{y}}(t)], \boldsymbol{K(t)} = \boldsymbol{P(t)H}$
  - Riccati Differential Equation(Covariance propagation and update)
    $\frac{\mathrm{d}\boldsymbol{P(t)}}{\mathrm{d}t} = \boldsymbol{F(t)P(t)} + \boldsymbol{P(t)F^T(t)} - \boldsymbol{P(t)H^T(t)R^{-1}(t)H(t)P(t)} + \boldsymbol{G(t)Q(t)G^T(t)}$
- Linearized Kalman Filter is simple,but performs poorly when the actual trajectory deviates from a nominal trajectory.The state transition and measurement are linear approximation, while the true system is nonlinear.

9.4 Extended Kalman Filter

- Extended Kalman Filter is a significant improvement in two major aspects:
  - The jacobian matrices are ecaluated not at nominal state $\mathbf{x}^*(\mathbf{t})$ but at an estimated state $\mathbf{x}(\mathbf{t})$

$$F(t) = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}(t)} \qquad F(t) = \left.\frac{\partial f}{\partial x}\right|_{x^*}$$

$$H(t) = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}(t)} \qquad H(t) = \left.\frac{\partial h}{\partial x}\right|_{x^*}$$

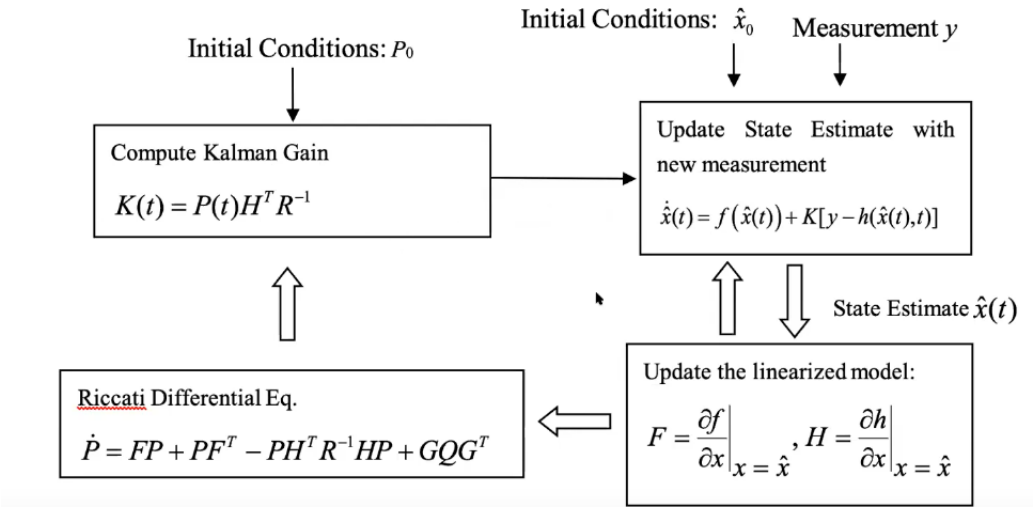


  - State propagation and update use the full nonlinear state equation and measurement equation.

$$\dot{\hat{x}} = f(\hat{x}(t), t) + K(t)[y(t) - h(\hat{x}(t), t)] \qquad \dot{\hat{x}} = F(t)\hat{x}(t) + K(t)[y(t) - H(t)\hat{x}(t)]$$

$$\hat{y}(t)$$

7



- Covariance propagation and update, however, are based on the linearized model of the nonlinear system. Namely, we use the Riccati Differential Equation using the Jacobian matrices evaluated at estimated state $\hat{\mathbf{x}}(\mathbf{t})$

$$\frac{dP(t)}{dt} = F(t)P(t) + P(t)F^T(t) - P(t)H^T(t)R^{-1}(t)H(t)P(t) + G(t)Q(t)G^T(t)$$

First-order approximation of nonlinear functions

- Extended Kalman Filter is a nonlinear Filter.
- Optimality of state estimation is no longer guaranteed.
- Extended Kalman. Filter(EKF) tends to underestimate the error covariance,when the system is highly nonlinear.This sometimes causes the divergence of estimate. Divergence Scenario:
  Underestimated P → Small Kalman Gain K → Insufficient State Update → Growing estimation error → Inaccurate Jacobians F(t) and H(t) → Blow up

9.5 Uncented Kalman Filter

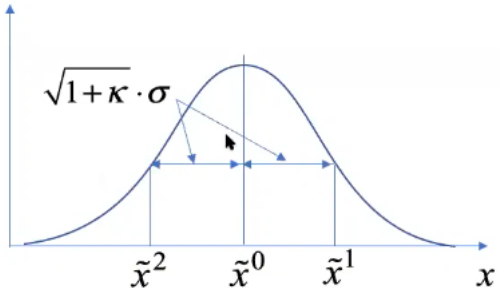
- Unscented Kalman Filter can better handle such a problem.
- Unscented Kalman Filter, originally developed by Julier and Uhlmann[1997], uses a different method for computing error covariance matrices.
- It does not use the Riccati Equations (continuous time)or the covariance propagation and update laws (discrete time).Instead it uses a special technique,called Unscented Transform,for propagating and updating covariance matrices.
- The key idea is to estimate the error covariance based on a special set of sample points,termed "sigma points",which propagate directly through the original nonlinear model.

9.6 Unscented Transform

- Consider a simple case where one-dimensional random variable X has a Gaussian distribution.
- Mean $\bar{\mathbf{x}}$ and σ^2 completely characterize the distribution .

- Unscented transform represents Gaussian distribution with three special sample points, called Sigma Points.

$$\tilde{x}^0 = \bar{x} \qquad W_0 = \frac{\kappa}{1+\kappa}$$
$$\tilde{x}^1 = \bar{x} + \sqrt{1+\kappa}\cdot\sigma \qquad W_1 = W_2 = \frac{1}{2(1+\kappa)}$$
$$\tilde{x}^2 = \bar{x} - \sqrt{1+\kappa}\cdot\sigma$$
$$p(x) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\exp\left(-\frac{(x-\bar{x})^2}{\sigma^2}\right)$$



and W_i is Sigma Points

where K is a parameter of sigma points to be tuned,and

9.7 **Example**

- Consider a piecewise linear function for $\mathbf{y} = \mathbf{g}(\mathbf{x})$, as a shown in the figure.(The two lines are connected with a smooth curve for differentiability.)
- The mean of x is transformed to $\bar{\mathbf{y}} = \mathbf{g}(\bar{\mathbf{x}})$, which is not the true mean of the distribution of y.We can see by inspection:

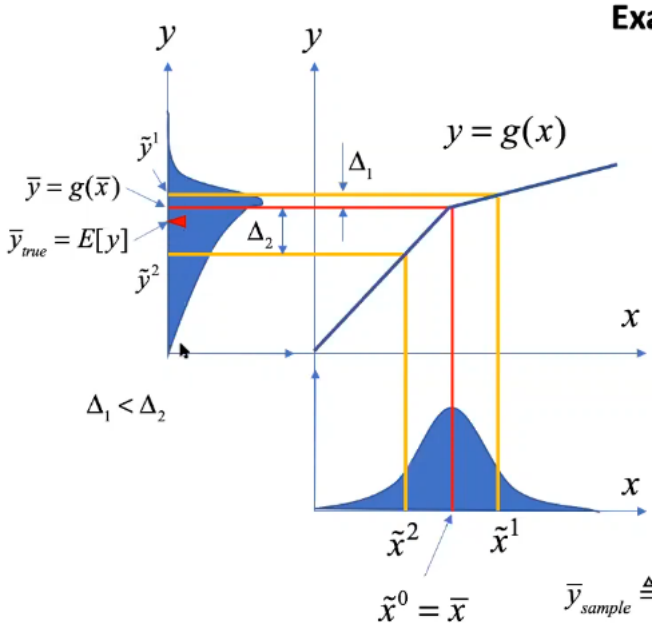
$$\bar{\mathbf{y}}_{\mathbf{ture}} = \mathbf{E}[\mathbf{y}] < \bar{\mathbf{y}} = \mathbf{g}(\bar{\mathbf{x}})$$

- The weighted mean of the Sigma points can give a better mean value.

$$\bar{y}_{sample} \triangleq \sum_{i=0}^{2} W_i \tilde{y}_i = \frac{k}{1+k}\tilde{y}_0 + \frac{1}{2(1+k)}(\tilde{y}_1 + \tilde{y}_2$$

$$= \frac{k}{1+k}\bar{y} + \frac{1}{2(1+k)}(\bar{y} + \Delta_1 + \bar{y} - \Delta_2) = \bar{y} + \frac{1}{2(1+k)}(\Delta_1 - \Delta_2) < \bar{y}, \quad \Delta_1 < \Delta$$

The weighted mean of the Sigma points gives a better mean value



9.8 Sigma Points for Multivariate Gaussian Distribution

- In general, for an n-dimensional Gaussian distribution,we use (2n+1) Sigma points.

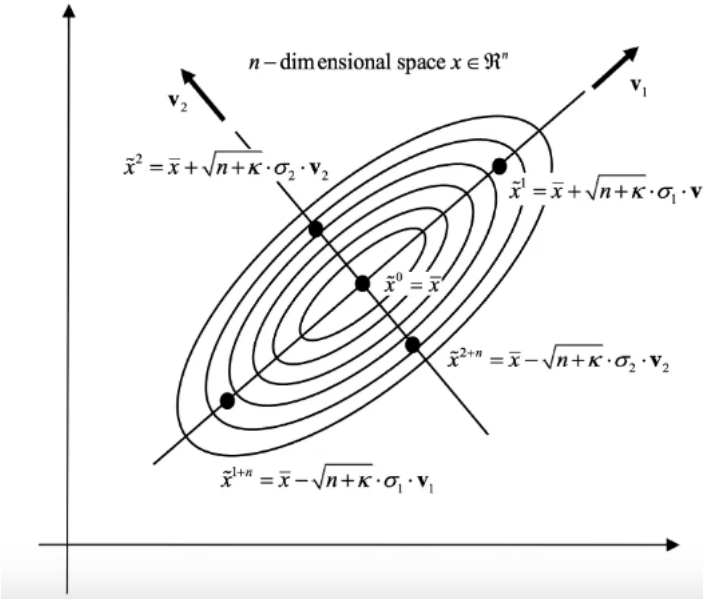
$$\mathbf{p(x) = det(2\pi P_x)^{-\frac{1}{2}} exp\left\{ -\frac{1}{2}(x-\bar{x})^T P_x^{-1}(x-\bar{x}) \right\}}$$

- Note that the covariance matrix P_x is real,symmetric, and positive-definite.Therefore,it can be diagonalized

$$\mathbf{P_x = VDV^T}$$

Where $\mathbf{V = (v_1 v_n), D = diag(\sigma_1^2 \sigma_n^2)}$

- Sigma points are taken along the individual eigen vectors with unit length,$v_1, \ldots, v_n$



$$\tilde{x}^0 = \bar{x} \qquad W_0 = \frac{\kappa}{n+\kappa}$$

$$\tilde{x}^i = \bar{x} + \sqrt{n+\kappa} \cdot \sigma_i \cdot \mathbf{v}_i \qquad W_i = W_{i+n} = \frac{1}{2(n+\kappa)}$$

$$\tilde{x}^{i+n} = \bar{x} - \sqrt{n+\kappa} \cdot \sigma_i \cdot \mathbf{v}_i \qquad i = 1, \cdots, n$$

- As before,all the Sigma points propagate through a nonlinear analytic function, y=g(x),which is multivariate.

$$\tilde{y}_i = g(\tilde{x}_i), \quad i = 0, \ldots, 2n$$

- Fore these propagated points,the weighted mean is computed:

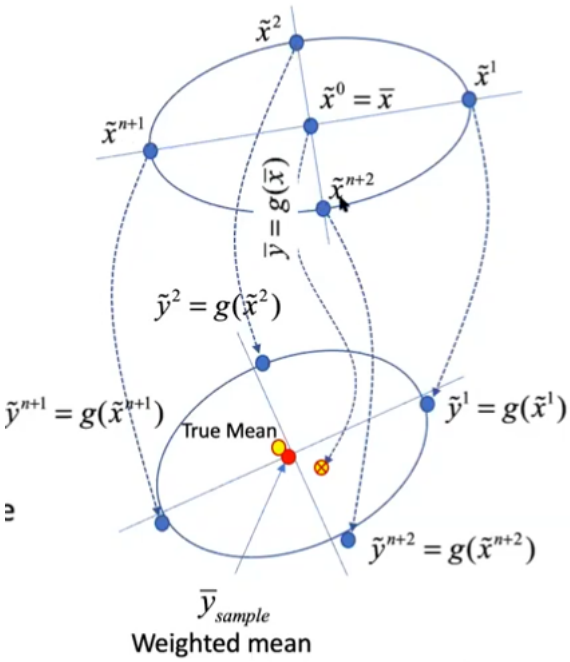
$$\tilde{y}_{sample} = \sum_{i=0}^{2n} W_i \tilde{y}_i$$

- The weighted covariance is given by

$$P_{y,sample)} = \sum_{t=0}^{2n} W_t(\tilde{y}_i - \bar{y}_{sample})(i - \tilde{\bar{y}_{sample}})^T$$

- We can show that the weighted mean can approximate the true mean to third order,and the weighted covariance to the second order.

- This sampling method is called **Unceed Transmor**



9.9 Propagation of State

- Find eigenvalues and eigen-vectors of covariance P_{t-1} and generate (2n+1) Sigma points

$$\tilde{x}_{t-1}^0 = \hat{x}_{t-1} : \quad \bar{x}$$

$$\tilde{x}_{t-1}^i = \hat{x}_{t-1} + \sqrt{n+\kappa} \cdot \sigma_i \cdot \mathbf{v}_i, \quad \tilde{x}_{t-1}^{i+n} = \hat{x}_{t-1} - \sqrt{n+\kappa} \cdot \sigma_i \cdot \mathbf{v}_i$$

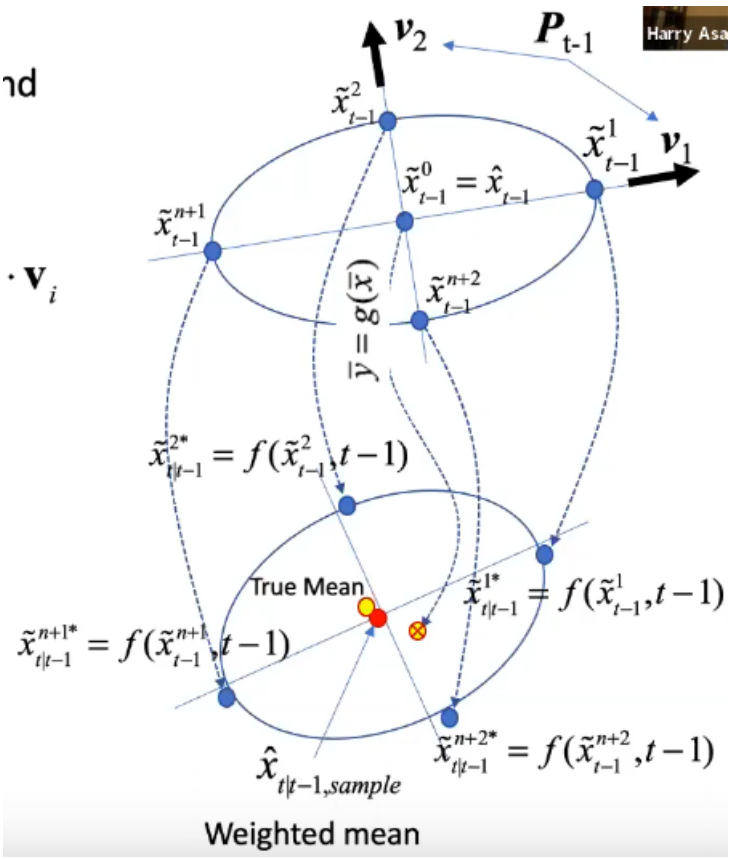
$$i = 1, \cdots, n$$

- Propagate the Sigma points through the state equation,noting that the process noise is zero mean.

$$\tilde{\mathbf{x}}_{\mathbf{t|t-1}}^{\mathbf{i}} = \mathbf{f}(\mathbf{t-1^i}, \mathbf{t-1}) + \tilde{\mathbf{w}_{\mathbf{t-1}}}, \quad \mathbf{i = 0, \ldots, 2n}$$

- For these (2n+1) Sigma points,the weighted mean is computed:

$$\hat{\mathbf{x}}_{\mathbf{t|t-1,sample}} = \sum_{\mathbf{i=0}}^{\mathbf{2}} \mathbf{W_i x_{t|\hat{t}-1}}$$



9.10 Propagation of Covariance

- a priori covariance is computed as
  $\mathbf{P_{t|t-1}} = \mathbf{E}[(\hat{\mathbf{x}}_{\mathbf{t|t-1}} - \mathbf{x_t})(\hat{\mathbf{x}}_{\mathbf{t|t+1}} - \mathbf{x_t})]$
  $= \mathbf{E}[(\hat{\mathbf{x}}_{\mathbf{t|t-1}} - \mathbf{f}(\mathbf{x_{t-1}}, \mathbf{t-1}) - \mathbf{w_{t-1}})(\hat{\mathbf{x}}_{\mathbf{t|t-1}} - \mathbf{f}(\mathbf{x_{t-1}}, \mathbf{t-1}) - \mathbf{w_{t-1}})^{\mathbf{T}}]$
  $= \mathbf{E}[(\hat{\mathbf{x}}_{\mathbf{t|t-1}}) - \mathbf{w_{t-1}})(\hat{\mathbf{x}}_{\mathbf{t|t-1}} - \mathbf{f}(\mathbf{x_{t-1}}, \mathbf{t-1}))^{\mathbf{T}}] + \mathbf{E}[\mathbf{w_{t-1}}\mathbf{w}^{\mathbf{T}}_{\mathbf{t-1}}] + (cross-terms)$
- The term $f(x_{t-1}, t-1)$ is not known since we do not know the exact state x_{t-1}.However,it is approximated to the $3^r d$ order with the weighted mean; $\hat{\mathbf{X}}_{\mathbf{t|t-1,sample}}$.There replace $f(x_{t-1}, t-1)$ by the weighted mean,and compute the a priori error covariance using the (2n+1) Sigma points.

$$\boldsymbol{P_{t|t-1,sample}} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{i=2n}} \boldsymbol{W_i}(\tilde{\boldsymbol{x}}^{\boldsymbol{i}}_{\boldsymbol{t|t-1}} - \hat{\boldsymbol{x}}_{\boldsymbol{t|t-1,sample}})(\tilde{\boldsymbol{x}}^{\boldsymbol{i}}_{\boldsymbol{t|t-1}} - \hat{\boldsymbol{x}}_{\boldsymbol{t|t-1,sample}})^{\boldsymbol{T}} + \boldsymbol{Q_{t-1}}$$

- Note that the predicted covariance is correct up to the second order.For brevity,the subscript "sample" will be dropped hereafter.

9.11 Obtaining the Kalman Gain from Innovation Covariance

- We aim to update state and covariance using Sigma points.The standard Kalman gain and covariance update laws,however,are not applicable to Unscented Transformation.Instead,we will consider an alternative method based on innovation.
- The Kalman Gain is given by $\mathbf{K_t} = \mathbf{P_{t|t-1}}\mathbf{H}^{\mathbf{T}}_{\mathbf{t}}(\mathbf{H_t}\mathbf{P}\mathbf{t}|\mathbf{t-1}\mathbf{H}^{\mathbf{T}}_{\mathbf{t}} + \mathbf{R_t})^{-\mathbf{1}}$, but the matrix $\mathbf{H_t}$ is not available for our nonlinear system.
- Instead,we use the following formula,

$$\mathbf{K_t} = \mathbf{P_{xy}}\mathbf{P}^{-\mathbf{1}}_{\mathbf{y}}$$

Where $\boldsymbol{P_y} = \boldsymbol{E}[(\boldsymbol{y_t} - \hat{\boldsymbol{y}}_{\boldsymbol{t}})(\boldsymbol{y_t} - \hat{\boldsymbol{y}}_{\boldsymbol{t}})^{\boldsymbol{T}}], \boldsymbol{P_{xy}} = \boldsymbol{E}[(\boldsymbol{x_t} - \hat{\boldsymbol{x}}_{\boldsymbol{t|t-1}}(\boldsymbol{y_t} - \hat{\boldsymbol{y}}_{\boldsymbol{t}})^{\boldsymbol{T}}]$
For a linear time-varying system,this new formula can be proven by construction.

$$
\begin{aligned}
P_y &\triangleq E\left[(y_t-\hat{y}_t)(y_t-\hat{y}_t)^T\right] \\
&= E\left[(H_t x_t + v_t - H_t\hat{x}_{t|t-1})(H_t x_t + v_t - H_t\hat{x}_{t|t-1})^T\right] \\
&= H_t E\left[(x_t-\hat{x}_{t|t-1})(x_t-\hat{x}_{t|t-1})^T\right]H_t^T + E\left[v_t v_t^T\right] \\
&= H_t P_{t|t-1} H_t^T + R_t
\end{aligned}
$$

$$
\begin{aligned}
P_{xy} &\triangleq E\left[(\hat{x}_{t|t-1}-x_t)(\hat{y}_t-y_t)^T\right] \\
&= E\left[(\hat{x}_{t|t-1}-x_t)(H_t\hat{x}_{t|t-1}-H_t x_t - v_t)^T\right] \\
&= E\left[(\hat{x}_{t|t-1}-x_t)(\hat{x}_{t|t-1}-x_t)^T\right]H_t^T - E\left[(\hat{x}_{t|t-1}-x_t\right. \\
&= P_{t|t-1}H_t^T
\end{aligned}
$$

Combining the above two,the new formula of the kalman gain is proven.The covariance of output y is called "Innovation Covariance".

9.12 State Update

- From the Innovation Covariance,

$$
\begin{aligned}
&\boldsymbol{P_y}[(\boldsymbol{y_t}-\boldsymbol{\hat{y}_t})(\boldsymbol{y_t}-\boldsymbol{\hat{y}_t})^{\boldsymbol{T}}] \\
&= \boldsymbol{E}[(\boldsymbol{h}(\boldsymbol{x_t},\boldsymbol{t})+\boldsymbol{v_t}-\boldsymbol{h}(\boldsymbol{\hat{x}_{t|t-1}},\boldsymbol{t}))(\boldsymbol{h}(\boldsymbol{x_t},\boldsymbol{t})+\boldsymbol{v_t}-\boldsymbol{h}(\boldsymbol{\hat{x}_{t|t-1}},\boldsymbol{t}))^{\boldsymbol{T}}] \\
&= \boldsymbol{E}[(\boldsymbol{h}(\boldsymbol{x_t},\boldsymbol{t})-\boldsymbol{h}(\boldsymbol{\hat{x}_{t|t-1}},\boldsymbol{t}))(\boldsymbol{h}(\boldsymbol{x_t},\boldsymbol{t})-\boldsymbol{h}(\boldsymbol{\hat{x}_{t|t-1}},\boldsymbol{t}))^{\boldsymbol{T}}] + \boldsymbol{E}[\boldsymbol{v_t}\boldsymbol{v_t^T}]
\end{aligned}
$$

- Computing the first term using Sigma points,

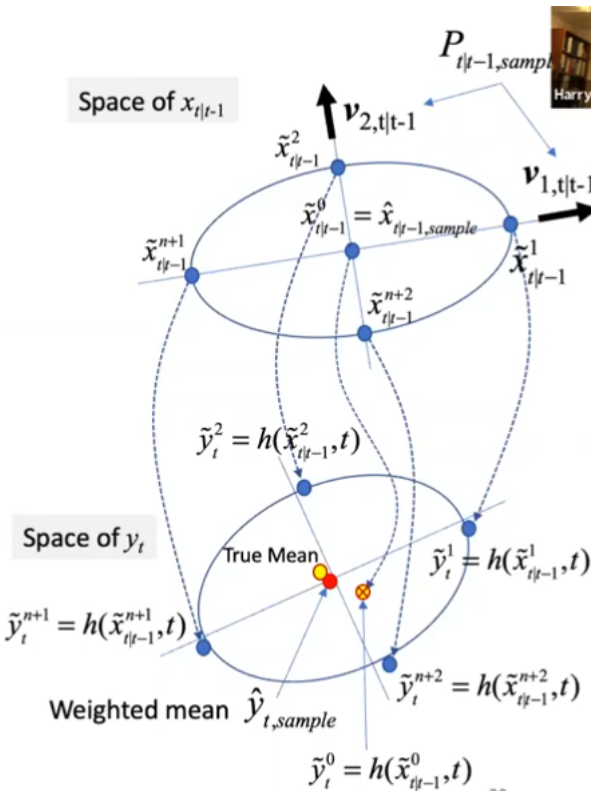
$$
\boldsymbol{P_y} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{2n}} \boldsymbol{W_i}(\boldsymbol{\tilde{y}_t^i}-\boldsymbol{\hat{y}_{t,sample}})(\boldsymbol{\tilde{y}_t^i}-\boldsymbol{\hat{y}_{t,sample}})^{\boldsymbol{T}} + \boldsymbol{R_t}
$$

- Similarly,the cross-covariance is given by

$$
\boldsymbol{P_{xy}} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{2n}} \boldsymbol{W_i}(\boldsymbol{\tilde{x}_{t|t-1}^i}-\boldsymbol{\hat{x}_{t|t-1,sample}})(\boldsymbol{\tilde{y}_t^i}-\boldsymbol{\hat{y}_{t,sample}})^{\boldsymbol{T}}
$$

- The Kalman Gain is then $\mathbf{K_t} = \mathbf{P_{xy}}\mathbf{P_y^{-1}}$
- The state update is given by

$$
\mathbf{\hat{x}_t} = \mathbf{\hat{x}_{t|t-1,sample}} + \mathbf{K_t}[\mathbf{y_t} - \mathbf{\hat{y}_{t,sample}}]
$$



9.13 Covariance Update

- The Covariance update formula includes measurement matrix H_p which must be replaced. We can use the innovation covariance for this:

$$\mathbf{P_y} = \mathbf{E}[(\mathbf{y_t} - \hat{\mathbf{y}}_\mathbf{t})(\mathbf{y_t} - \hat{\mathbf{y}}_\mathbf{t})^\mathbf{T}]$$

- Insering $P_y P_y^{-1} = I$ in the covariance update formula,

$$\mathbf{P_t} = (\mathbf{I} - \mathbf{K_t H_t})\mathbf{P_{t|t-1}} = \mathbf{P_{t|t-1}} - \mathbf{K_t H_t P_{t|t-1}}$$

$$= \mathbf{P_{t|t-1}} - \mathbf{K_t P_y P_y^{-1} H_t P_{t|t-1}} = \mathbf{P_{t|t-1}} - \mathbf{K_t P_y K_t^T}$$

  Where the Kalman gain $\mathbf{K_t} = \mathbf{P_{t|t-1} H_t^T P_y^{-1}}$ is used to eliminate the measurement matrix H_t

- Using the Sigma points,this can be computed as follows.

$$\mathbf{P_t} \cong \mathbf{P_{t|t-1.sample}} - \mathbf{K_t P_y K_t^T}$$

9.14 The Recursive Algotithm of Unscented Kalman Filter

- Given $_{t-1}$ and $\mathbf{P_{t-1}}$,sample sigma points by computing eigenvalues and eigenvector of P_{t-1};
- Propagate the sigma points through the nonlinear model to obtain $\tilde{\mathbf{x}}^\mathbf{i}_{\mathbf{t|t-1}} = \mathbf{f}(\tilde{\mathbf{x}}^\mathbf{i}_{\mathbf{t-1}}, \mathbf{t-1})$;
- From the (2n+1) sigma points compute the mean and variance:

$$\hat{\boldsymbol{x}}_{\boldsymbol{t|t-1,sample}} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{2n}} \boldsymbol{W_i} \hat{\boldsymbol{x}}^{\boldsymbol{i}}_{\boldsymbol{t|t-1}}$$

$$\boldsymbol{P_{t|t-1,sample}} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{i=2n}} \boldsymbol{W_i}(\boldsymbol{t|t-1}^{\boldsymbol{i}} - \hat{\boldsymbol{x}}_{\boldsymbol{t|t-1,sample}})(\boldsymbol{t|t-\tilde{1}}^{\boldsymbol{i}} - \hat{\boldsymbol{x}}_{\boldsymbol{t|t-1,\tilde{sample}}})^{\boldsymbol{T}} + \boldsymbol{Q_t - 1}$$

- Sample again (2n+1) sigma points for $P_{t|t-1,sample}$
- Transform the propagated sigma points to output estimate $\mathbf{y^i_t}$ based on the nonlinear measurement equation, and compute the estimated output

$$\hat{\boldsymbol{y}}_{\boldsymbol{t,sample}} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{2n}} \boldsymbol{W_i} \tilde{\boldsymbol{y}}^{\boldsymbol{i}}_{\boldsymbol{t}}, \quad \tilde{\boldsymbol{y}}^{\boldsymbol{i}}_{\boldsymbol{t}} = \boldsymbol{h}(\tilde{\boldsymbol{x}}^{\boldsymbol{i}}_{\boldsymbol{t|t-1}}, \boldsymbol{t})$$

- Evalute the innovation covariance and the cross covariance by using (2n+1) points of propagated output estimates to find the Kalman gain

$$\mathbf{K_t} = \mathbf{P_{xy} P_y^{-1}}$$

$$\boldsymbol{P_y} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{2n}} \boldsymbol{W_i}(\tilde{\boldsymbol{y}}^{\boldsymbol{i}}_{\boldsymbol{t}} - \hat{\boldsymbol{y}}_{\boldsymbol{t,sample}})(\tilde{\boldsymbol{y}}^{\boldsymbol{i}}_{\boldsymbol{t}} - \hat{\boldsymbol{y}}_{\boldsymbol{t,sample}})^{\boldsymbol{T}} + \boldsymbol{R_t}$$

$$\boldsymbol{P_{xy}} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{i=2n}} \boldsymbol{W_i}(\boldsymbol{t|t-1}^{\boldsymbol{i}} - \hat{\boldsymbol{x}}_{\boldsymbol{t|t-1,\tilde{sample}}})(\tilde{\boldsymbol{y}}^{\boldsymbol{i}}_{\boldsymbol{t}} - \hat{\boldsymbol{y}}_{\boldsymbol{t,sample}})^{\boldsymbol{T}}$$

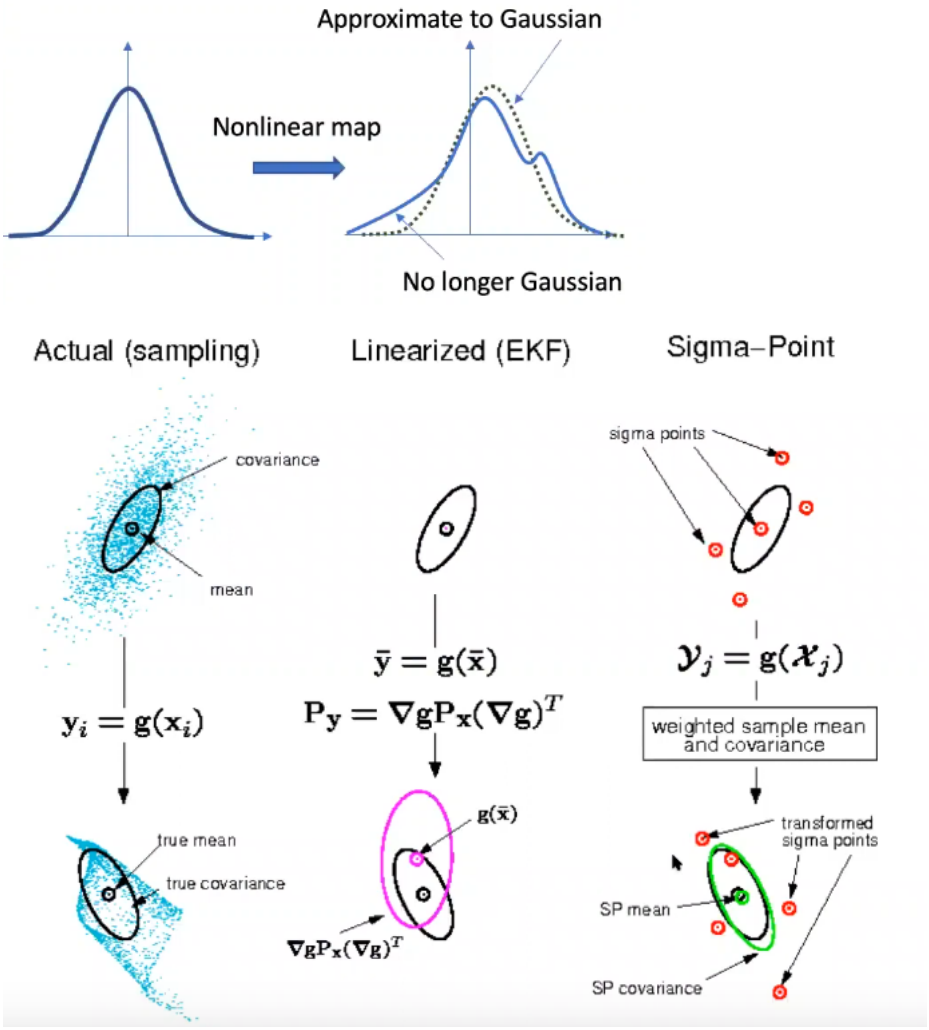
- Update the state estimate with the Kalman gain;

$$\hat{\mathbf{x}}_\mathbf{t} = \hat{\mathbf{x}}_{\mathbf{t|t-1,sample}+\mathbf{K_t}[\mathbf{y_t} - \hat{\mathbf{y}}_{\mathbf{t,sample}}]}$$

- Update the a posteriori covariance;

$$\mathbf{P_t} \cong \mathbf{P_{t|t-1,sample}} - \mathbf{K_t P_y K_t^T}$$

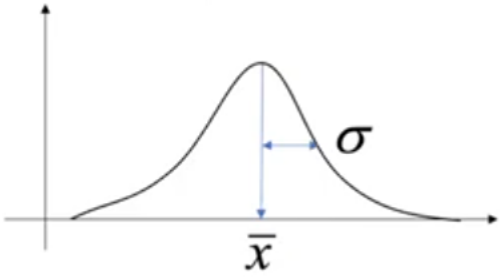
- Set t=t+1, and repeat the above process.
- No jacobian,no partial derivatives are needed.
- The estimated covariance using sigma points is more accurate than the jacobian-based one.
- **Caveat!** The distribution of random variables after transformed through nonlinear equations,e.g.f(x,t),h(x,t),is no longer Gaussian, although the original distribution was Gassian. Unscented Kalman Filter approximates this distirbution to a Gaussian and characterizes whith mean and covariance.Although this approximation is accurate to the 2^{nd} order,the discrepancy from a complete Gaussian may grow,as the process is repeated.



- EKf tends to underestimate covariance $\mathbf{P_t}$ and thereby uses a smaller Kalman gain.
- This leads to an insufficient correction (update) to the state estimation.
- Such an niaccurate state estimation further incurs inaccurate covariance and the situation may smallball,leading to a divergence.
- UKF is more reliable particularly when covariance rapidly change.

10. **Bayes Filter**

❑ Giving a particular value as estimate makes sense when the state distribution is Gaussian or unimodal.

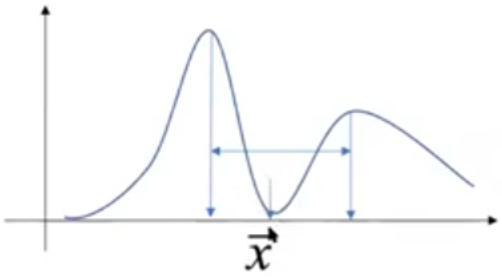


Gaussian Distribution

Mean $\bar{x}$ and standard deviation σ

Mean represents the estimated value very well, with Standard Deviation being Accuracy of prediction.

❑ However, if the distribution is not Gaussian and multimodal, the single value is a poor representation.



Non-Gaussian Distribution

A mean value does not represent the overall distribution of the random variables.
The mean is the least likely value in this example.

3

Belief

Non-Gaussian Distribution

A mean value does not represent the overall distribution of the random variables.

Belief $g(x)$: the entire pdf distribution rather than a single value.

Bayes Filter predicts the pdf distribution of a random variable.

$$\begin{array}{ccccccc} \hat{x}_{t-1} & \mapsto & \hat{x}_{t|t-1} & \mapsto & \hat{x}_t & & \text{Kalman Filter} \\ g_{t-1}(x_{t-1}) & \mapsto & g_{t|t-1}(x_t) & \mapsto & g_{t|t}(x_t) & & \text{Bayes Filter} \end{array}$$

Propagate Update

at time t-1 a priori estimate. a posteriori estimate

10.1 Markov Process

- Discrete-time stochastic state transitiion,in general:

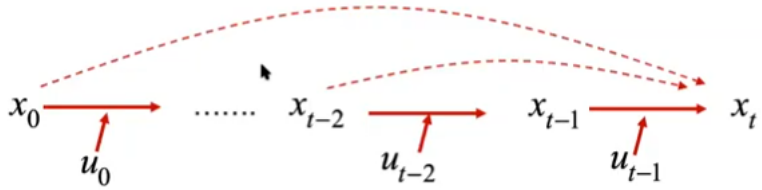
$$\mathbf{Pr}(\mathbf{x_t}|\mathbf{x_0}, \ldots, \mathbf{x_{t-1}}, \mathbf{u_0}, \ldots, \mathbf{u_{t-1}})$$

The probability of random variable $\mathbf{X_t} = \mathbf{x_t}$, given previous state and input $x_0, \ldots, x_{t-1}, u_0, \ldots, u_{t-1}$

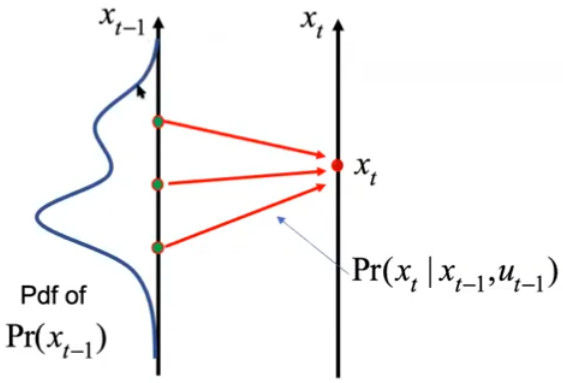
- A special case where the probability of $X_t = x_t$ depends only on x_{t-1}, u_{t-1}, the

process is called a Markov Process.

$$\mathbf{Pr(x_t|x_{t-1}, u_{t-1}}$$



10.2 Champman-Kolmogorov Equation



State $X_t = x_t$ can be reached from x_{t-1} with conditional probability density of

$$\Pr(x_t \mid x_{t-1}, u_{t-1})$$

where $-\infty < x_{t-1} < \infty$ has a pdf of $\Pr(x_{t-1})$

Therefore, the pdf of $\Pr(x_t)$ is given by

$$\Pr(x_t) = \int_{-\infty}^{\infty} \Pr(x_t \mid x_{t-1}, u_{t-1}) \Pr(x_{t-1})\, dx_{t-1}$$

This is called the Chapman-Kolmogorov Equation.

10.3 State Propagation Law

- In our Bayes Filter problem,we want to recursively estimate a priori belief $\mathbf{g_{t|t-1}(x_t)}$ from a posterior belief at time t-1,$g_{t-1}(x_{t-1})$
- Given a state transition equation

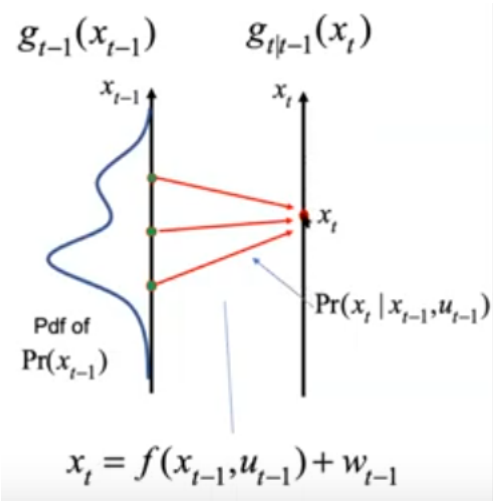
$$\mathbf{x_t} = \mathbf{f(x_{t-1}, u_{t-1})} + \mathbf{w_{t-1}}$$

$\mathbf{w_{t-1}}$:Additive noise

- Applying the Chapman-Kolmogorov Equation,

$$\mathbf{g_{t|t-1}(x_t)} = \int_{-\infty}^{\infty} \mathbf{Pr(x_t|x_{t-1}, u_{t-1})g_{t-1}(x_{t-1})dx_{t-1}}$$

How can we find this probability?

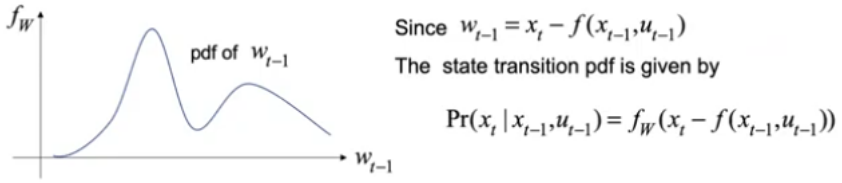


The State Transition Equation

$$\mathbf{x_t} = \mathbf{f}(\mathbf{x_{t-1}}, \mathbf{u_{t-1}}) + \mathbf{w_{t-1}}$$

. Deterministic Random

Given x_{t-1} and u_{t-1}, the randomness of x_t comes from process noise $w_{t-1} \backsim f_W(w_{t-1})$



Back to the Chapman-Kolmogorov equation:

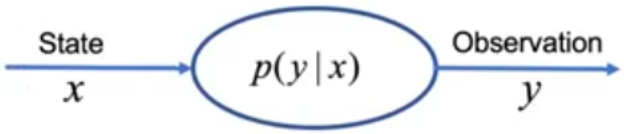
$$\boldsymbol{g_{t|t-1}(x_t) = \int_{-\infty}^{\infty} f_W(x_t - f(x_{t-1}, u_{t-1}))g_{t-1}(x_{t-1}dx_{t-1})}$$

10.4 Bayes'Rule

- Joint probability density

$$\mathbf{p(x, y) = p(x|y)p(y) = p(y|x)p(x)}$$

$$\mathbf{\therefore p(x|y) = \frac{p(y|x)p(x)}{p(y)}}$$



- Suppose that we know the conditional density p(y—x) how can we estimate the state x from observation y:p(x—y)?
- Remark: p(y)=$\int_{-\infty}^{\infty} p(y|x)p(x)dx$

$$\left(\mathbf{1 = \int_{-\infty}^{\infty} p(x|y)dx = \int_{-\infty}^{\infty} \frac{p(y|x)p(x)}{p(y)}dx = \frac{1}{p(y)}\int_{-\infty}^{\infty} p(y|x)p(x)dx}\right)$$

- Therefore, denominator p(y) is a scalling factor.
  p(y) is nothing but a scaling factor that makes $\boldsymbol{\int_{-\infty}^{\infty} p(x|y)dx = 1}$

- We do not need to know p(y).So, let's replace it by a constant: $\boldsymbol{p(y) = \frac{1}{\eta}}$

10.5 The Bayes Filter Algorithm

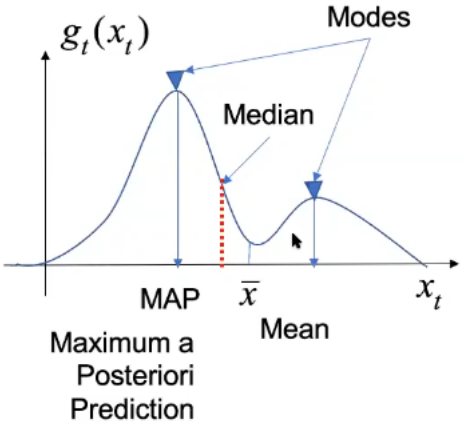
- Iniital Conditions: $\mathbf{g_0(x_0)}$ set t=1;
- Belief Propagation:

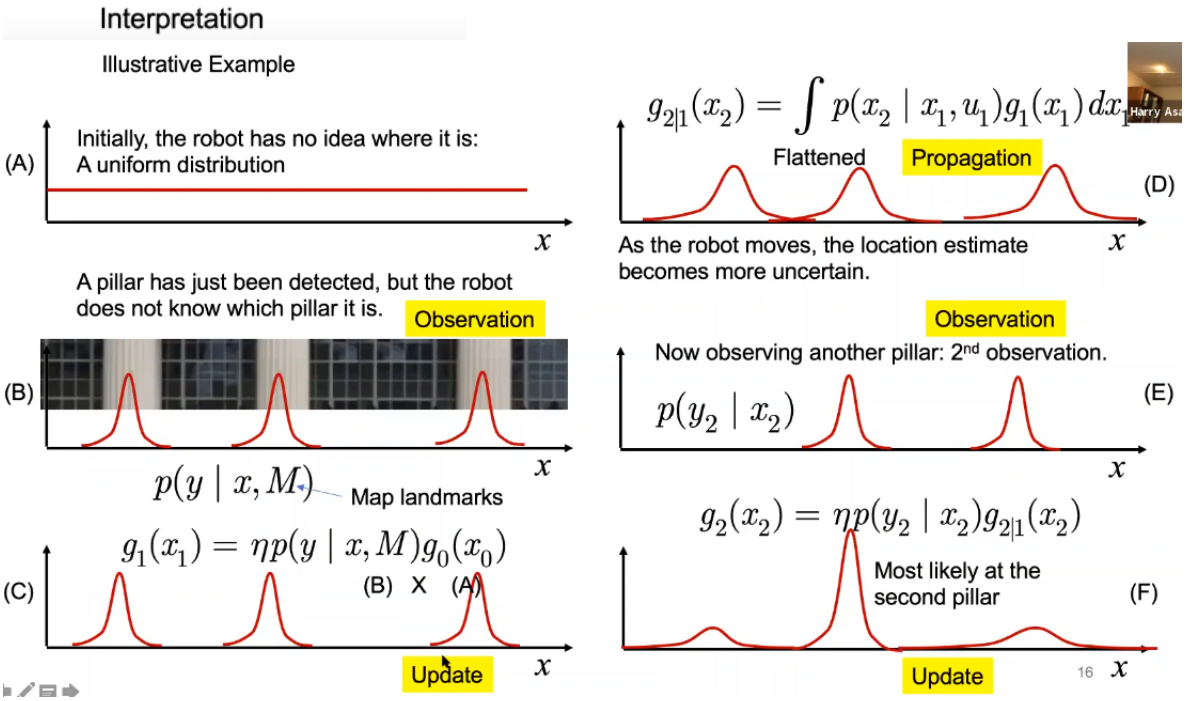
$$\boldsymbol{g_{t|t-1}(x_t) = \int_{-\infty}^{\infty} f_W(x_t - f(x_{t-1}, u_{t-1}))g_{t-1}(x_{t-1}dx_{t-1}}$$

- Assimilate y_t and update the a priori belief

$$\boldsymbol{g_t(x_t) = \eta f_V(y_t - h(x_t, t))g_{t|t-1}(x_t)}$$

- Return $g_t(x_t)$. Set t=t+1 and repeat
  Interpretation of the belief $\mathbf{g_t(x_t)}$.
  - Maximum a Posteriori Prediction (MAP)
  - Modes (multiple)
  - Medium
  - Mean





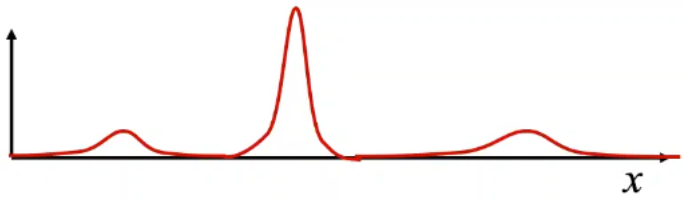
10.6 Bayes Filter

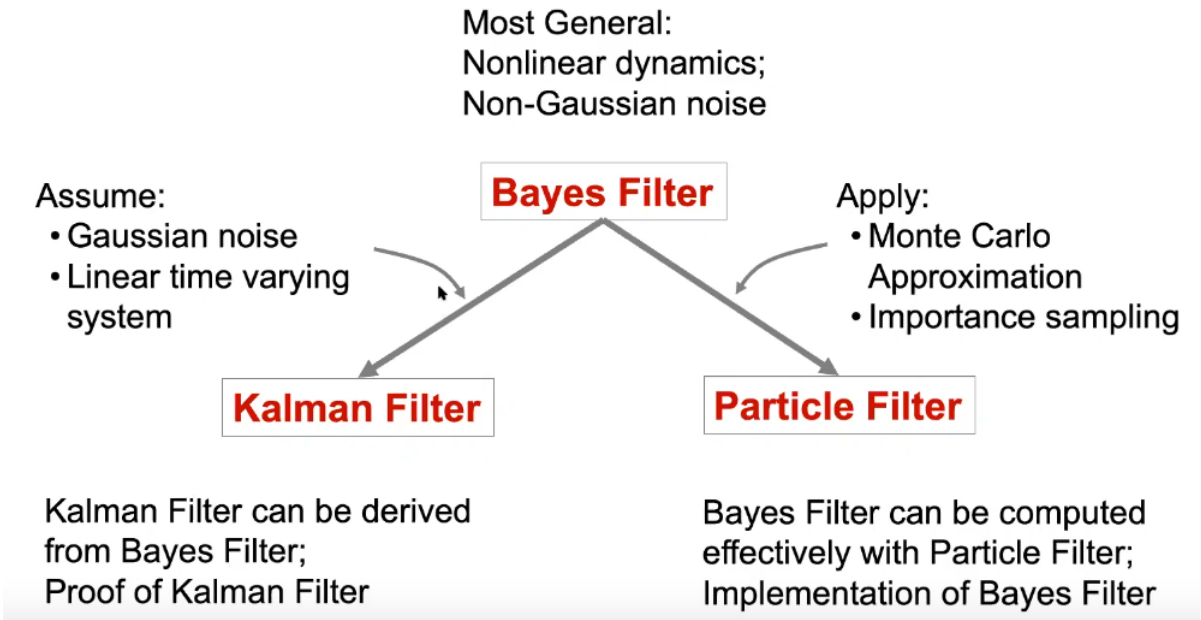
- Nonlinear dynamics, non-Gaussian distribution

- Use of "Belief", pdf estimation rather than a single value estimation.
- Bimodal,skewed distribution of state can be treated.
- Multiple Hypothesis Tracking: all possible casees are tracked.
- Low probability cases, too, are not eliminated.
- Computationally expensive:

$$g_{t|t-1}(x_t) = \int_{-\infty}^{\infty} f_W(x_t - f(x_{t-1}, u_{t-1}))g_{t-1}(x_{t-1}dx_{t-1}$$

$$g_t(x_t) = \eta f_V(y_t - h(x,t))g_{t|t-1}(x_t)$$





10.7 Gaussian Kalman Filter (so-called Kalman Filter)

- Kalman Filter is a special case of Bayes Filter.We can derive Kalman Filter from Bayes Filter by making the following assumptions.
- Assume a linear Time-Varying stochastic system:

$$\mathbf{x_{t+1}} = \mathbf{A_t x_t} + \mathbf{B_t u_t} + \mathbf{G_t w_t}$$

Further assume for brevity $\mathbf{u_t} = \mathbf{0}, \mathbf{G_t} = \mathbf{0}$

$$\mathbf{y_t} = \mathbf{H_t x_t} + \mathbf{v_t}$$

- Assume white (uncorrelated) Gaussian noise:

$$\mathbf{w_t} \sim \mathbf{N}(\mathbf{0}, \mathbf{Q_t}), \mathbf{v_t} \sim \mathbf{N}(\mathbf{0}, \mathbf{R_t})$$

Where

where

$$E[w_t w_s^T] = \begin{cases} Q_t; & t = s \\ 0; & t \neq s \end{cases}, \quad E[v_t v_s^T] = \begin{cases} R_t; & t = s \\ 0; & t \neq s \end{cases}, \quad E[w_t v_s^T] = 0, \forall t, \forall s$$

and Gaussian distribution

$$f_W = \frac{1}{\sqrt{\det(2\pi Q_t)}} \exp\left(-\frac{1}{2} w_t^T Q_t^{-1} w_t\right), \quad f_V = \frac{1}{\sqrt{\det(2\pi R_t)}} \exp\left(-\frac{1}{2} v_t^T R_t^{-1} v_t\right)$$

10.8 Deriving Kalman Filter from Bayes Filter

- Goal:Find an optimal state estimate that minimizes the mean squared prediction error conditioned by all the prior observations and inputs.

$$\hat{\mathbf{x}}_\mathbf{t}^\mathbf{o} = \arg\min_{\hat{\mathbf{x}}_\mathbf{t}} \mathbf{E}[|\hat{\mathbf{x}}_\mathbf{t} - \mathbf{x}_\mathbf{t}|^\mathbf{2} | \mathbf{y_1}, \ldots, \mathbf{y_t}, \mathbf{u_1}, \ldots, \mathbf{u_{t-1}}]$$

- This is equivalent to the conditional mean:

$$\hat{\mathbf{x}}_\mathbf{t}^\mathbf{o} = \mathbf{E}[\hat{\mathbf{x}}_\mathbf{t} | \mathbf{y_1}, \ldots, \mathbf{y_t}]$$

Check this for a scalar case:

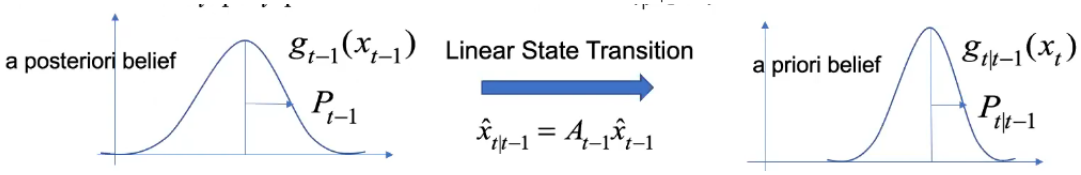
$$\mathbf{J} = \mathbf{E}[|\hat{\mathbf{x}} - \mathbf{x}|^\mathbf{2}]$$

$$\frac{\mathbf{d}J}{\mathbf{d}\hat{x}} = 2E[\hat{x} - x] = 2(\hat{x} - E[x]) = 0, \therefore \hat{x} = E[x]$$

10.9 Proof of Gaussian Kalman Filter(Outline)

Step1

- Given that $\mathbf{g_{t-1}(x_{t-1})}$ is Gaussian, show that $\mathbf{g_{t|t-1}(x_t)}$, too, is Gaussian.



- Use Induction: if the distribution of $\mathbf{x_{t-1}}$ is Gaussian with mena $\hat{\mathbf{x}}_\mathbf{t-1}$ and covariance $\mathbf{P_{t-1}}$

$$g_{t-1}(x_{t-1}) = \frac{1}{\sqrt{\det(2\pi P_{t-1})}} \exp\left(-\frac{1}{2}(x_{t-1} - \hat{x}_{t-1})^T P_{t-1}^{-1}(x_{t-1} - \hat{x}_{t-1})\right)$$

Then we can show that $g_{t|t-1}(x_t)$ is also Gaussian.

$$g_{t|t-1}(x_t) = \frac{1}{\sqrt{\det(2\pi P_{t|t-1})}} \exp\left(-\frac{1}{2}(x_t - \hat{x}_{t|t-1})^T P_{t|t-1}^{-1}(x_t - \hat{x}_{t|t-1})\right)$$

where $P_{t|t-1} = A_{t-1} P_{t-1} A_{t-1}^T + Q_{t-1}$

Linear state propagation does not distort the distribution.

- This is a highly technical derivation.See the lecture notes for details.

Step2:Brief Update

- Recall $\boldsymbol{g_t(x_t) = \eta f_V(y_t - h(x_t, t)) g_{t|t-1}(x_t)}$
  - $\mathbf{f_V}$=Measurement noise pdf
  - $\mathbf{exp}() : \mathbf{h(x_t, t)}$=Gaussian with Covariance R_t
  - $\mathbf{exp}() : \mathbf{g_{t|t-1}}$=Gaussian with Covariance P_{t-1}

– Therefore, the belief update should be in the following form:

$$\mathbf{g_t(x_t) = \grave{\eta} exp[]exp[] = \grave{n} exp[+] = \grave{\eta} exp[-N(x_t)]}$$

– Here,

$$\mathbf{N(x_t) = \frac{1}{2}(y_t - H_t x_t)^T R_t^{-1}(y_t - H_t x_t) + \frac{1}{2}(x_t - \hat{x}_{t|t-1})^T P_{t|t-1}^{-1}(x_t - \hat{x}_{t|t-1}}$$

– This is a quadratic function of x_t

$$\mathbf{N(x_t) = \frac{1}{2} x_t^T (H_t^T R_t^{-1} H_t + P_{t|t-1}^{-1}) x_t + \dots}$$

$\frac{dN(x_t)}{dx_t} = 0$
Where $N(x_t) = \frac{1}{2}(y_t - H_t x_t)^T R_t^{-1}(y_t - H_t x_t) + \frac{1}{2}(x_t - \hat{x}_{t|t-1})^T P_{t|t-1}^{-1}(x_t - \hat{x}_{t|t-1}$

– Recall $\frac{d}{dx} = 0 \rightarrow Ax + Bx = 0$
Therefore $\frac{dN(x_t)}{dx_t} = 0 \rightarrow \ -H_t^T R_t^{-1}(y_t - H_t x_t) + P_{t|t-1}^{-1}(x_t - \hat{x}_{t|t-1} = 0$

– Denoting x_t that satisfies the above optimality condition by $\hat{x}_t$

$$\mathbf{P_{t|t-1}^{-1}(\hat{x}_t - \hat{x}_{t|t-1}) = H_t^T R_t^{-1}(y_t - H_t \hat{x}_t + H_t \hat{x}_{t|t-1} - H_t \hat{x}_{t|t-1})}$$

$$\mathbf{= H_t^T R_t^{-1}(y_t - H_t \hat{x}_{t|t-1}) - H_t^T R_t^{-1} H_t(\hat{x}_t - \hat{x}_{t|t-1})}$$

$$\mathbf{\rightarrow (P_{t|t-1}^{-1} + H_t^T R_t^{-1} H_t(\hat{x}_t - \hat{x}_{t|t-1}) = H_t^T R_t^{-1}(y_t - H_t \hat{x}_{t|t-1})}$$

– Noting that $\mathbf{P_t^{-1} = P_{t|t-1}^{-1} + H_t^T R_t^{-1} H_t}$ and pre-multiplying P_t
$\therefore \mathbf{\hat{x}_t = \hat{x}_{t|t-1}^{-1} + P_t H_t^T R_t^{-1}(y_t - H_t \hat{x}_{t|t-1})}$ This agrees with the state update formula

Punch Line

- We have arrived at the familiar linear filter:

$$\mathbf{\hat{x}_t = \hat{x}_{t|t-1} + K_t(y_t - H_t \hat{x}_{t|t-1}), \quad K_t = P_t H_t^T R_t^{-1}}$$

- In this proof we have never assumed that the optimal filter is linear. Instead,the linear filter has been derived from the optimality conditions.
- Kalman Filter is optimal among linear and nonlinear filters, as long as the noise,w_t and v_t are Gaussian, and the process is linerar time-varying.

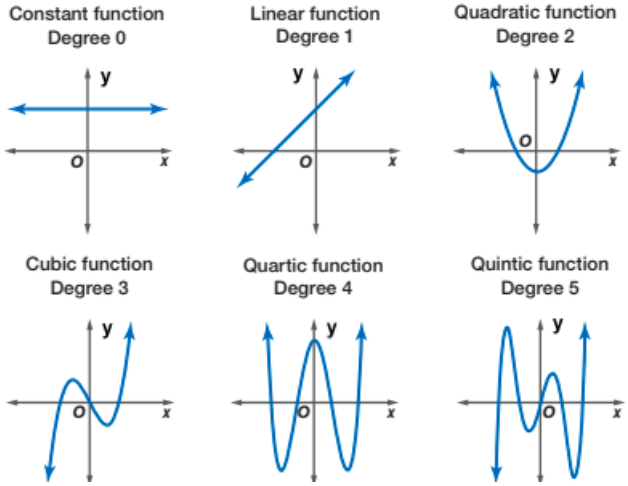
IV Path Planning

1. **Polynomials**
   - A Polynomial is a sum of Variable ruised to powers and multiplied by coefficient.

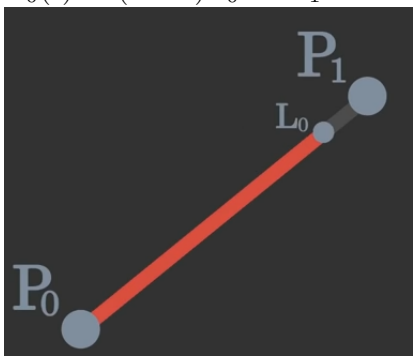
$$\sum_{i=0}^{n} a_i x^n = a_0 + a_1 x + a_2 x^2 + \_\_\_ + a_n x^n$$

   - The degree(or order of a polynomial is the highest power of the variable.

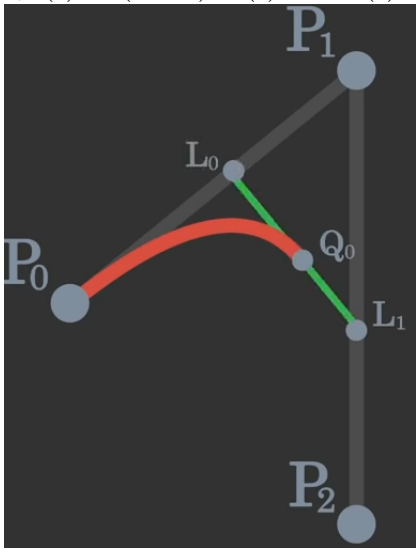
- Polynomials of different degrees are given names
  $f(x) = x + 1$*Linear*
  $g(x) = x^2 + x + 1$*Quadratic*
  $h(x) = x^3 + x^2 + x + 1$*Cubic*
  $i(x) = x^4 + x^3 + x^2 + x + 1$*Quartic*



2. **Beizier-path**
   +*Linear*
   $L_0(t) = (1-t)P_0 + tP_1$

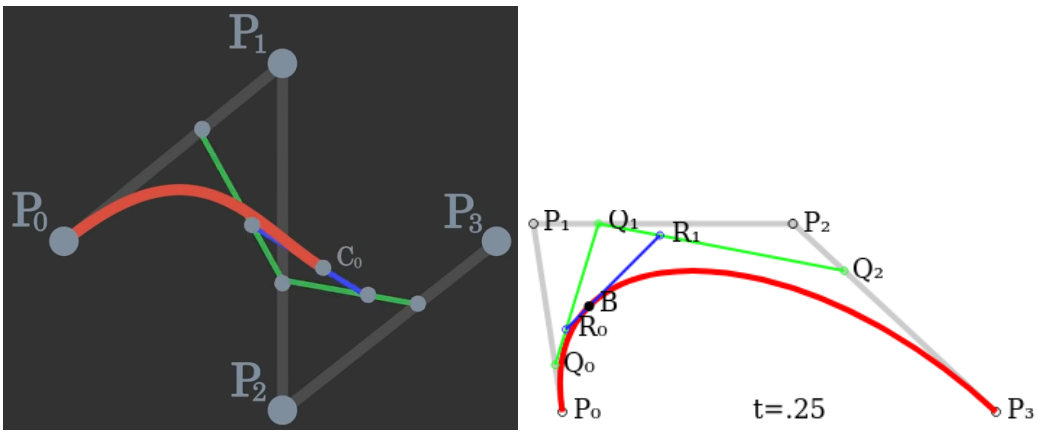


   +*Quadratic*
   $L_1(t) = (1-t)P_1 + tP_2$
   $Q_0(t) = (1-t)L_0(t) + tL_1(t) = (1-t)^2P_0 + 2(1-t)tP_1 + t^2P_2$



   +*Cubic*
   $L_2(t) = (1-t)P_2 + tP_3$

$Q_1(t) = (1-t)L_1(t) + tL_2(t)$
$C_0(t) = (1-t)Q_0(t) + tQ_1(t) = (1-t)^3P_0 + 3(1-t)^2tP_1 + 3(1-t)t^2P_2 + t^3P_3$
Where t varies from 0 to 1



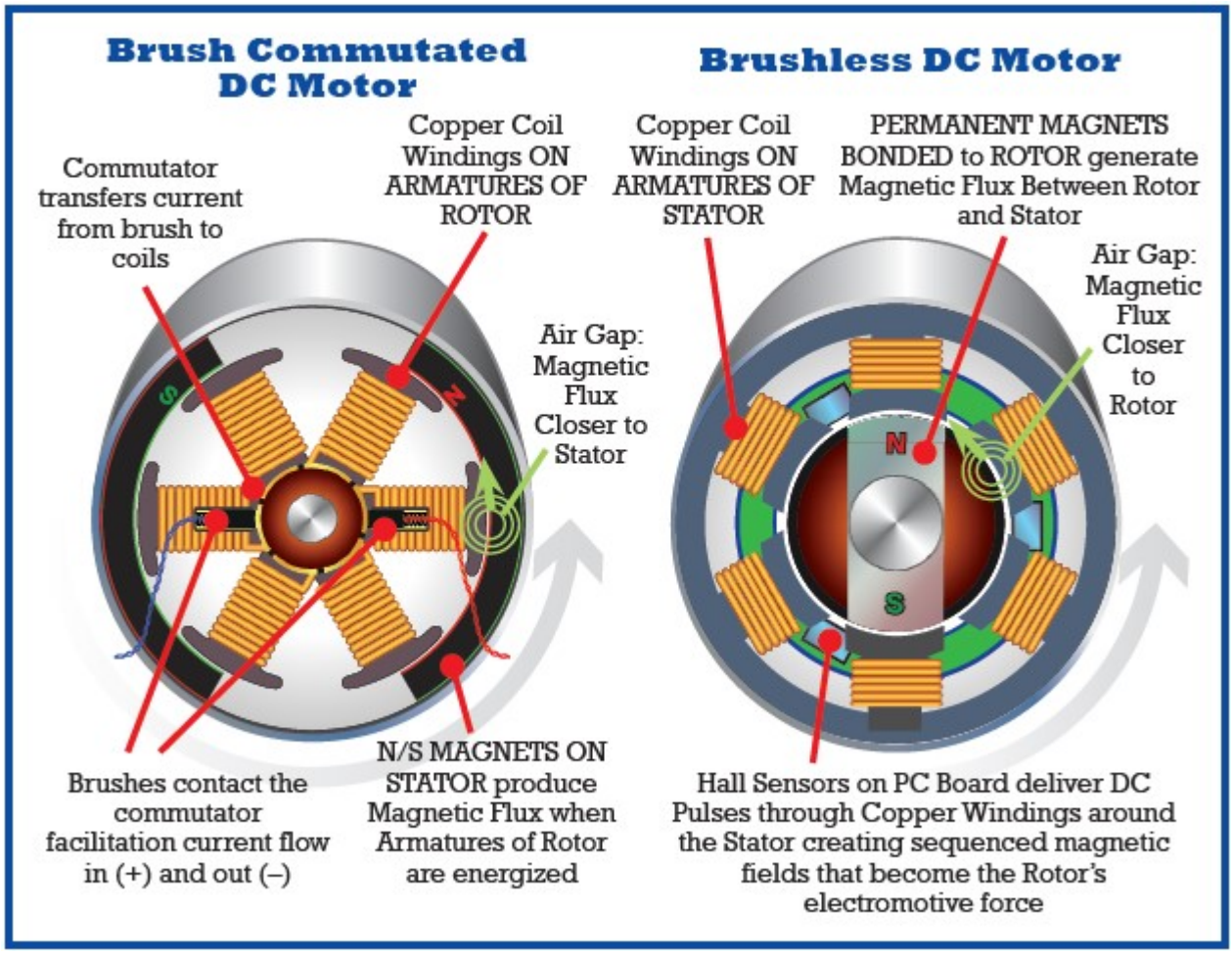
(a) (b)

V Modeling or Actuator

1. **DC_Motor**

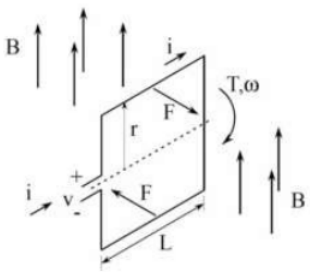
   1.1 **What is a DC_Motor?**

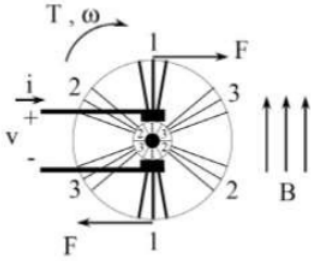
   A DC motor is defined as a class of electrical motors that convert direct current electrical energy into mechanical energy. From the above definition, we can conclude that any electric motor that is operated using direct current or DC is called a DC motor.



1.2 Control System Design for DC motor

DC motor: equations of motion in matrix form



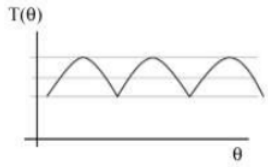


$$\begin{bmatrix} v_e \\ i \end{bmatrix} = \begin{bmatrix} 2BNlr & 0 \\ 0 & \frac{1}{2BNlr} \end{bmatrix} \begin{bmatrix} \omega \\ T \end{bmatrix}$$

or

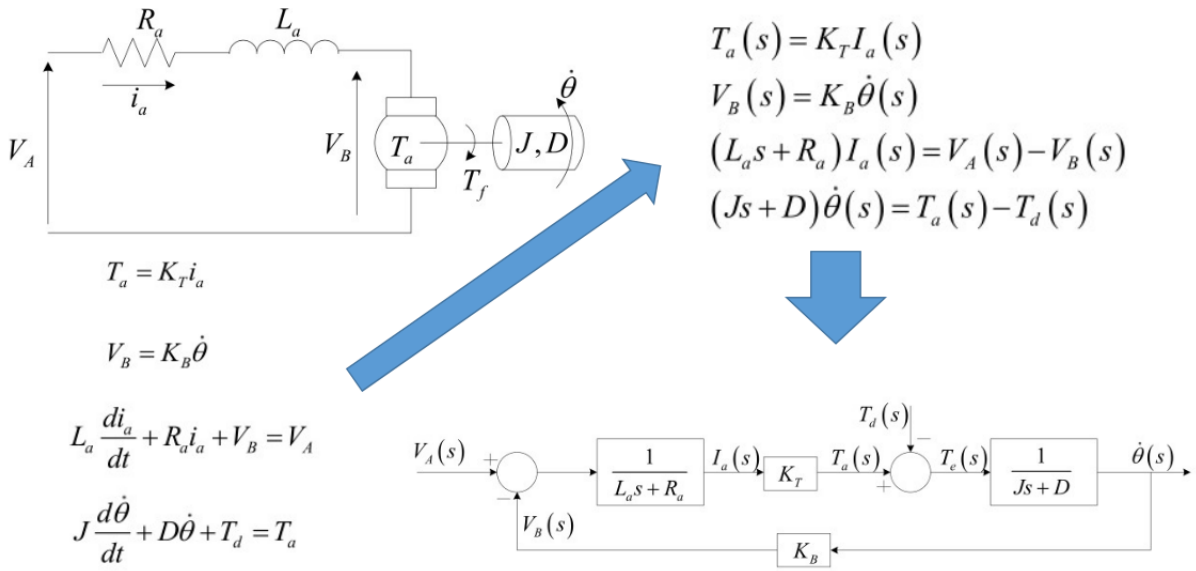
$$\begin{bmatrix} v_e \\ i \end{bmatrix} = \begin{bmatrix} K_v & 0 \\ 0 & \frac{1}{K_m} \end{bmatrix} \begin{bmatrix} \omega \\ T \end{bmatrix}$$

multiple windings *N*: continuity of torque



$i \equiv i_a, \qquad T \equiv T_a$

$v_e \equiv V_B, \qquad \dot{\theta} = \omega$



1.3 How to Simulation of DC motor?

For simulation during development.if our focus is not the performance of the speed,or position control of the DC motor,we can use simplified model ($L_a = 0, T_{fric} = D\omega$) as:

$$\tau\dot{\omega}(t) + \omega = Ku(t)$$

where u(t)=V_A, τ is called time constant.The above equation can be rewritten in Laplace transform as:

$$\frac{\Omega(s)}{U(s)} = \frac{K}{\tau s + 1}$$

If u(t) is step function,then $U(s) = \frac{V_0}{s}$,and we obtain

$$\Omega(s) = \frac{K}{\tau s + 1} U(s) = \frac{K}{\tau s + 1}\frac{V_0}{s} = KV_0\left(\frac{1}{s} - \frac{1}{\tau s + 1}\right) = KV_0\left(\frac{1}{s} - \frac{\tau}{s + 1/\tau}\right)$$

Therefore,$\omega(t) = KV_0(1 - e^{-t/\tau})$.For steady-state, $\omega_{ss} -> KV_0$

1.4 DC Motor System Identification

Transfer Function : $\left\{\begin{array}{l} \frac{\omega(s)}{V_m} = \frac{K_t}{(Js+b)(Ls+R_m)-K_tK_b} \\ \frac{\omega(s)}{T_l(s)} = \frac{Ls+R_m}{(Js+b)(Ls+R_m)-K_tK_b} \end{array}\right\}$

Since the time constant in electrical part $\tau_e = \frac{L}{R}$ too small compare to mechanical time constant.

we can write transfer function of DC motor a

$$Speed : H(s) = \frac{\omega(s)}{V_m} = \frac{K}{\tau s + 1}$$

$$Position : G(s) = \frac{\theta(s)}{V(s)} = \frac{K}{(\tau s + 1)s}$$

By performing experiments on a DC-motor,some of its physical parameters will be identified (also called estimated),and a first-order differential equation describing the dynamical behavior of the motor will be developed.As an alternative approach,a model is also developed.

using a simple step-response experiment:

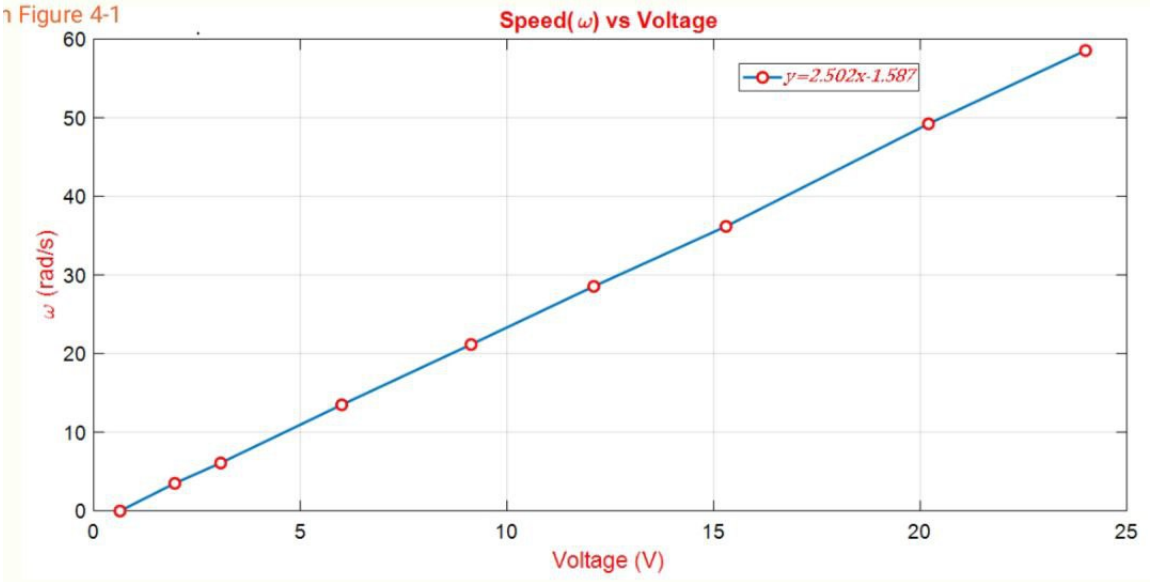
$$Transfer function : H(s) = \frac{\omega(s)}{V_m} = \frac{K}{\tau s + 1}$$

Where K:DC gain and τ:time constant
Identification parameter K and τ in our dc
Give the input voltag motor to DC motor and measure the speed ω at steady state below:

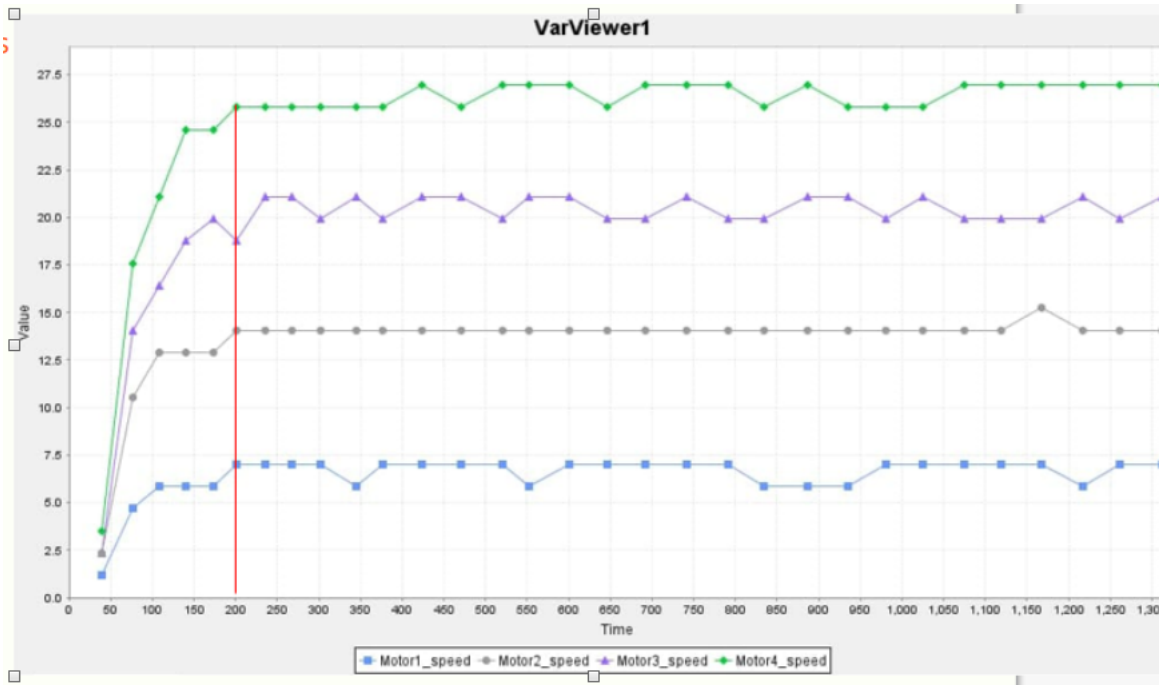
V(v)	1.96	3.07	6	9.13	12.1	15.3	20.2	24
ω_s(rad/s)	3.51	6.1	13.5	21.18	28.57	36.2	49.25	58.58

A line of best fit(Least Square Method)is a straight line that is the best approximation of the given set of data.

Figure 4-1



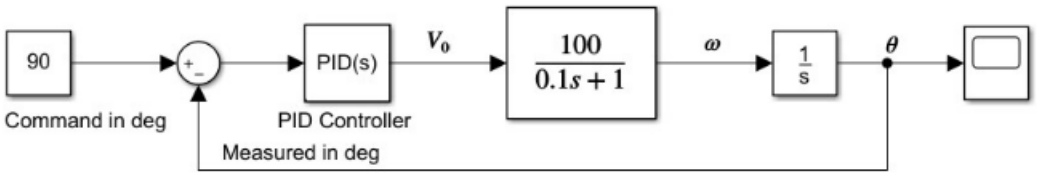
Since,the line equation **y=ax+b** that fitted the input voltage to output speed motor: y=2.502x-1.587
So K=2.502 (Rad/s)V
_Find τ time constant we have to measure trasient response in picture below.In this part we the input 3.5,6.5,8.5,10.4V ,and measure the trasient state.
_By take time $\tau = 0.2s$,setting time t=5τ => $\tau = 0.2/5 = 0.04$



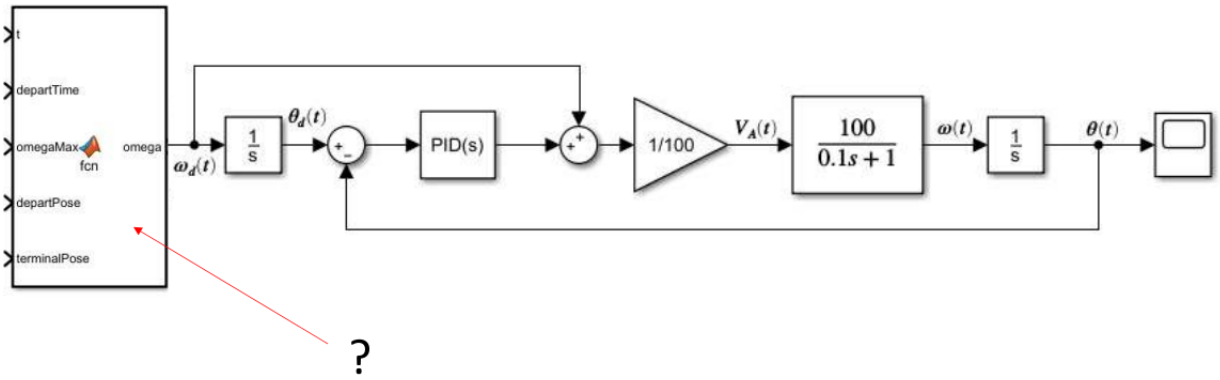
$$H(s) = \frac{\omega(s)}{V(s)} = \frac{2.502}{0.04s + 1}$$

1.5 PID_control DC_motor

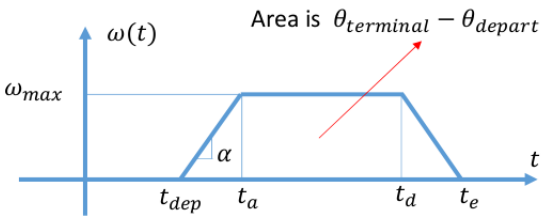
- **Position control**



- **Velocity Feedforward Position Feedback**

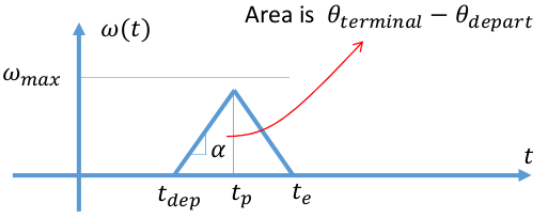


- **Velocity generation (given** $\alpha, \omega_{max}, \theta_{depart}, \theta_{terminal}, t_{dep}$)



Case 1: $t_a - t_{dep} = \frac{\omega_{max}}{\alpha}$,

$$t_d - t_{dep} = \frac{\theta_{terminal} - \theta_{depart} - \frac{\omega_{max}^2}{\alpha}}{\omega_{max}} + t_a$$

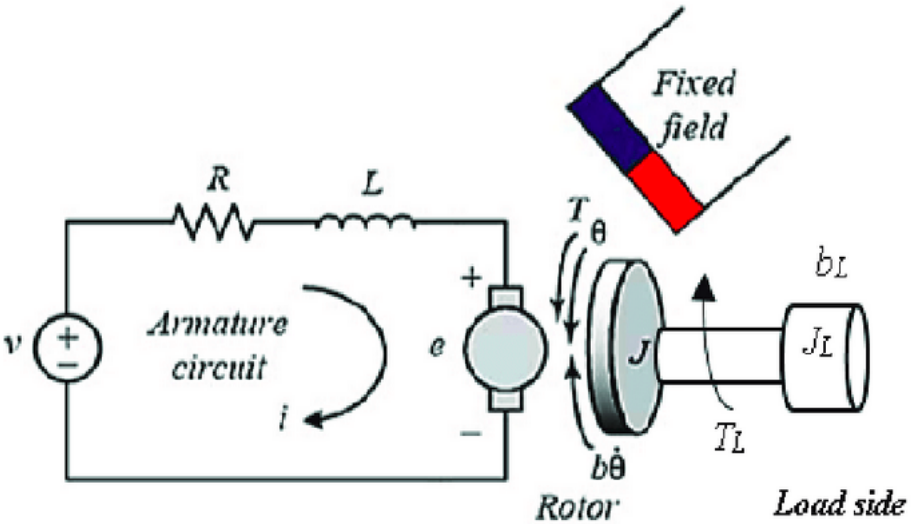


Case 2: $t_p - t_{dep} = \sqrt{\frac{\theta_{terminal} - \theta_{depart}}{\alpha}}$,

Condition that separate Case 1 and 2 is that at time t_p , the speed exactly reaches ω_{max}.

1.6 Step for find modeling of DC motor

- **Method1:**
  Transfer function form:H(s)=$\frac{\omega(s)}{V(s)} = \frac{K}{\tau s+1}$
  _Step1: you can Input Voltage(2.5 to Maximum voltage between 10 step) to find line equation y=ax+b graph_form(Voltage,$\omega(rad/s)$)
  _Step2: you take a=K .
  _Step3: How to find time constant τ
  _63.2% of Input final value. or
  _to measure trasient reponse.(you input voltage 4 time).
- **Method2:**
  _Step1:calculate only on part Electric



$$V_A = L_a\frac{di}{dt} + R_a i_a + K_B\omega)$$

_where L_a, R_a and K_B are constant
At steady state : ω_{ss}=const,i_a=const, then $\frac{di_a}{dt} = 0$ Hence,(1) becomes

$$V_A = K_B\omega + R_a i_a$$

_Step2:Measurable variableV_A, ω and i_a
You input Voltage($V_A = 5, 6, 7, 8, 10, 12, 14, 17, 20, 24V$) to measure i_a and ω
_Step3:Identifying Ra and K_B for a DC motor

We have $V_A = K_B\omega + R_a i_a = \begin{bmatrix}\omega & i_a\end{bmatrix}\begin{bmatrix}K_B \\ i_a\end{bmatrix}$

$$\begin{bmatrix}\omega_1 & i_1\\ \omega_2 & i_2\\ \omega_3 & i_3\\ \omega_4 & i_4\\ \omega_5 & i_5\\ \omega_6 & i_6\\ \omega_7 & i_7\\ \omega_8 & i_8\\ \omega_9 & i_9\end{bmatrix}\begin{bmatrix}K_B \\ R_a\end{bmatrix} = \begin{bmatrix}V_1\\ V_2\\ V_3\\ V_4\\ V_5\\ V_6\\ V_7\\ V_8\\ V_9\end{bmatrix}$$

Ax=B

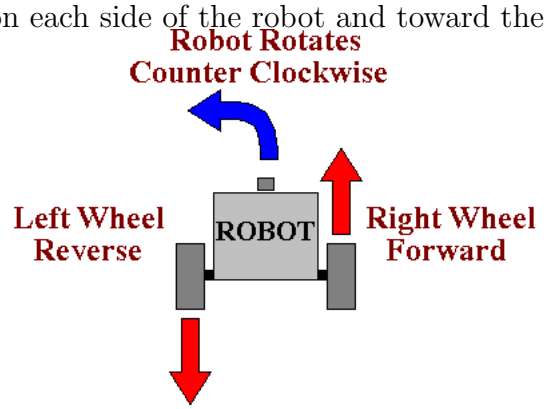
Then $x = \begin{bmatrix}K_B \\ R_a\end{bmatrix} = (A^T A)^{-1} A^T b$

2. **Wheel of mobile_robot**

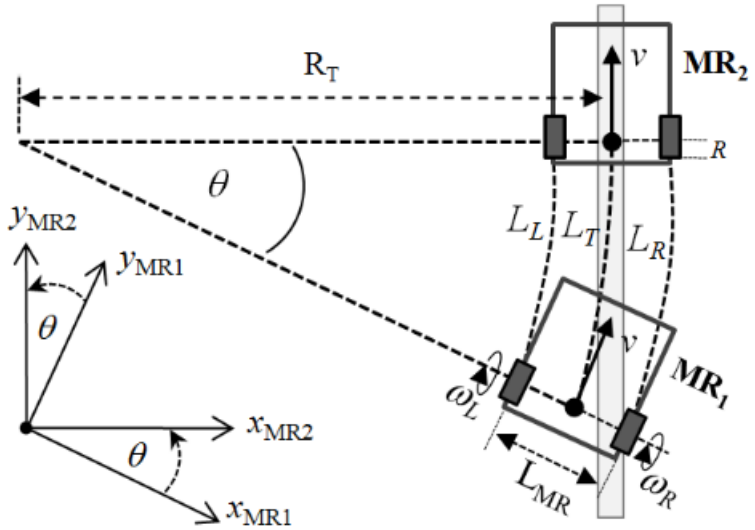
2.1 **Differential drive_Wheel:**

2.1.1 **What is Differential drive Wheel?**

The differential drive is a two-wheeled drive system with independent actuators for each wheel.The name refers to the fact that the motion vector of the robot is the sum of the independent wheel motions. The drive wheels are usually placed on each side of the robot and toward the front.



2.1.2 **Mathematical Model**



Linear velocities both of rear driven wheels V_R and V_L: can be calculated refer to the wheel radius R and angular velocity ω each wheel and:

$$\mathbf{V_R} = \omega_\mathbf{R}\mathbf{R}$$

$$\mathbf{V_L} = \omega_\mathbf{L}\mathbf{R}$$

A linear velocity V for the mobile robot Calculated by average of the angular velocity of right and left driven wheel:

$$\mathbf{V} = \frac{\mathbf{V_l} + \mathbf{V_r}}{\mathbf{2}}$$

Component of linear velocities $\dot{x}$ and $\dot{y}$ derived from the coordinate system consist of x and y position:

$$\dot{\mathbf{x}} = \mathbf{V}\mathbf{sin}(\dot{\theta})$$

$$\dot{\mathbf{y}} = \mathbf{V}\mathbf{cos}(\dot{\theta})$$

$$\dot{\theta} = \theta$$

Since the robot coordinate system are $[x_{MR1}, y_{MR1}]$ for the first mobile robot position, and $[x_{MR2}, y_{MR2}]$ for the second mobile robot position, the angle mobile robot orientation θ is the different between y_{MR2} and y_{MR1}.
A track curvature or length of track line the robot movement L_T is a function of left arc L_l and rigth L_R:

$$\mathbf{L_T} = \frac{\mathbf{L_R} + \mathbf{L_L}}{\mathbf{2}}$$

While the angle of orientation4 θ was linear to the (L_T, L_R, L_L) divided by radius of movement R_T,
The θ can be described to be:

$$\theta = \frac{\mathbf{L_T}}{\mathbf{R_T}}$$

$$\theta = \frac{\mathbf{L_R}}{\mathbf{R_T} + \frac{\mathbf{L_{MR}}}{\mathbf{2}}}$$

$$\theta = \frac{\mathbf{L_L}}{\mathbf{R_T} - \frac{\mathbf{L_{MR}}}{\mathbf{2}}}$$

The relation of θ to length L_l, L_R and distance of two-driven wheel L_{MR} is determined by:

$$\theta = \frac{\mathbf{L_R} - \mathbf{L_L}}{\mathbf{L_{MR}}}$$

Angular velocity $\dot{\theta}$ of the mobile robot turning is the different both of right velocity V_R and left linear velocity V_l to the distance of two driven wheels axis L_{MR}:

$$\theta = \frac{\mathbf{V_R} - \mathbf{V_L}}{\mathbf{L_{MR}}} = \frac{\omega_\mathbf{R} - \omega_\mathbf{L}}{\mathbf{L_{MR}}}\mathbf{R}$$

_The angular velocity of the both right and left wheel,can be reconstuct by derivation of

$$\omega_{\mathbf{R}} = (\mathbf{1} + \frac{\mathbf{L_{MR}}}{\mathbf{2R_T}})\frac{\mathbf{V}}{\mathbf{R}}$$

$$\omega_{\mathbf{L}} = (\mathbf{1} - \frac{\mathbf{L_{MR}}}{\mathbf{2R_T}})\frac{\mathbf{V}}{\mathbf{R}}$$

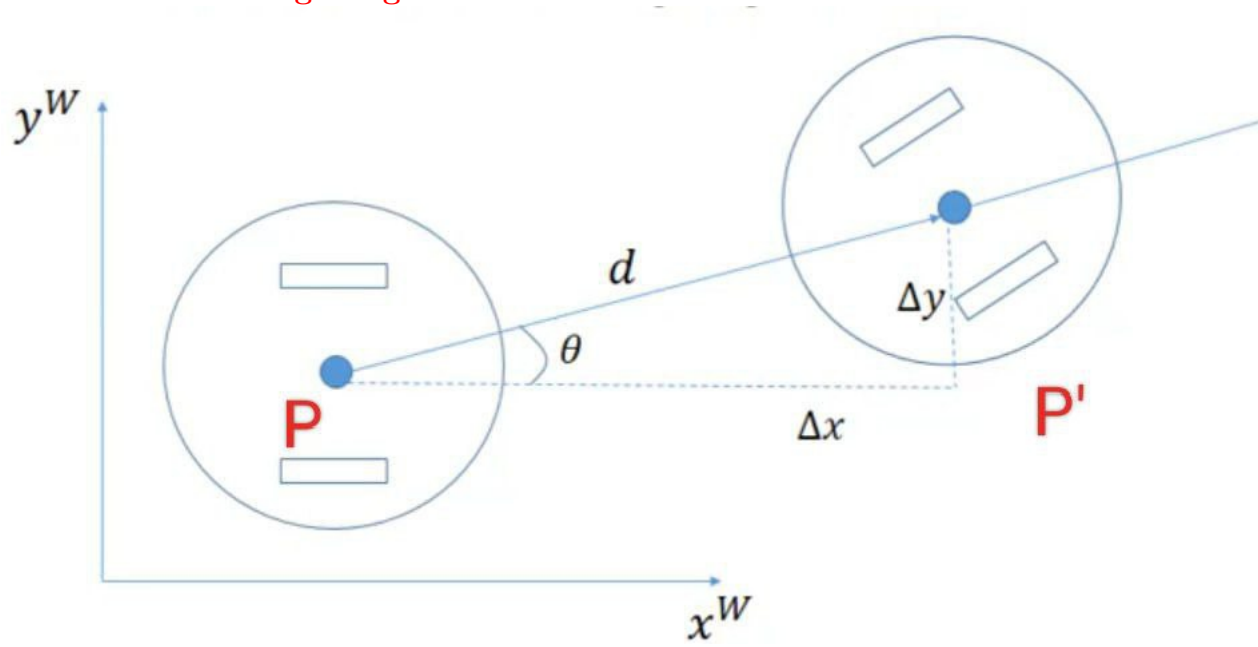
2.1.3 A Wheel Encoder track how much a wheel turns

C=wheel circumference:distance traveled in one revolution
q=number of encoder ticks per revolution
S_{it} =Encoder tick for wheel i at time t

$$\mathbf{L_l} = \frac{\mathbf{S_{Lt}} - \mathbf{S_{L(t-1)}}}{\mathbf{q}}\mathbf{C}$$

$$\mathbf{L_R} = \frac{\mathbf{S_{Rt}} - \mathbf{S_{R(t-1)}}}{\mathbf{q}}\mathbf{C}$$

2.1.4 Calculate the change in global coordinates

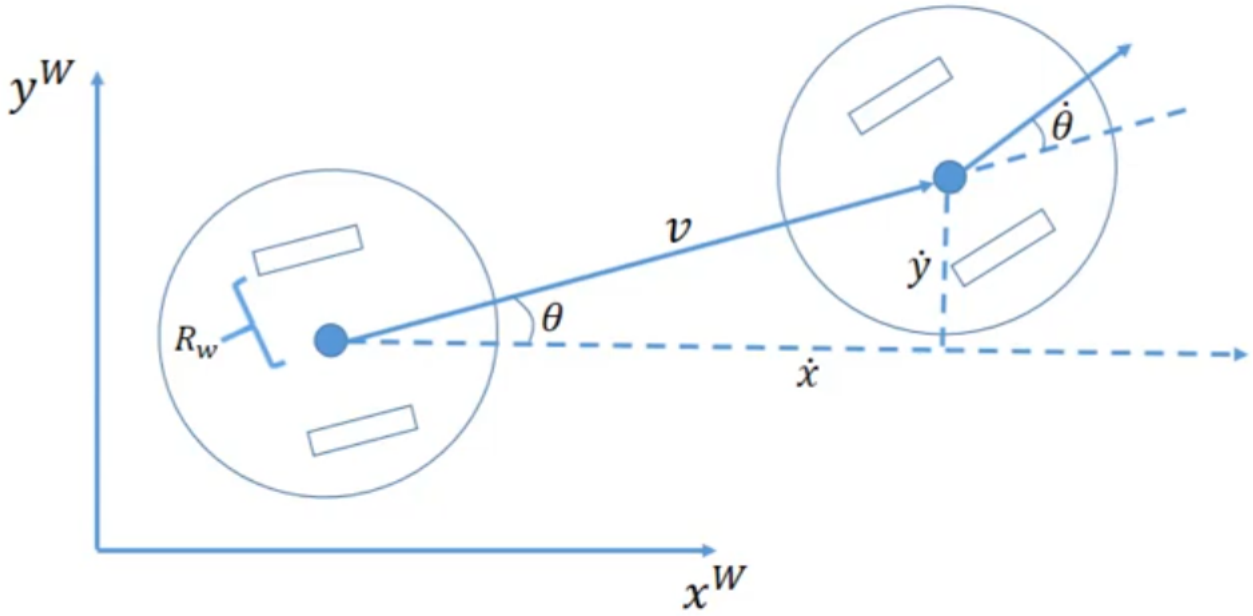


$\triangle\mathbf{x} = \mathbf{dcos}(\theta), \triangle\mathbf{y} = \mathbf{dsin}(\theta)$

$$\mathbf{P'} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} dcos(\theta) \\ dsin(\theta) \\ \triangle\theta \end{bmatrix}$$

Where: $\triangle\theta = \frac{\mathbf{d_r - d_l}}{\mathbf{2R_W}}$, $\mathbf{d} = \frac{\mathbf{d_l - d_r}}{\mathbf{2}}$

2.1.5 Calculate Velocity



$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}Vcos(\theta)\\ Vsin(\theta)\\ \frac{r(V_r-V_l}{2R_w}\end{bmatrix}$$

Where: $\frac{V_l+V_r}{2}$ and $\dot{\theta} = \frac{r}{2R_w}(V_r - V_l)$

_r=wheel radius

2.1.6 Kinematic of Wheel Differential in program

2.1.6. C/C++ Program

- **Forward Kinematic**

$$\mathbf{V_x} = \mathbf{Vsin}(\dot{\boldsymbol{\theta}})$$
$$\mathbf{V_y} = \mathbf{Vcos}(\dot{\boldsymbol{\theta}})$$
$$\dot{\boldsymbol{\theta}} = \boldsymbol{\theta}$$

- **Invers Kinematic**

_For Manual:Input V_x&ω

$$\boldsymbol{\omega_r} = \frac{\mathbf{2}}{\boldsymbol{R}}(\boldsymbol{V_x} + \boldsymbol{L} * \boldsymbol{\omega})$$

$$\boldsymbol{\omega_l} = \frac{\mathbf{2}}{\boldsymbol{R}}(\boldsymbol{V_x} - \boldsymbol{L} * \boldsymbol{\omega})$$

Where: ω_r:Angular Velocity of Right Wheel, ω_l:Angular Velocity of Left Wheel, ω:angular of robot:yaw

R=Radius of Wheel, L=Length from wheel right to wheel left.

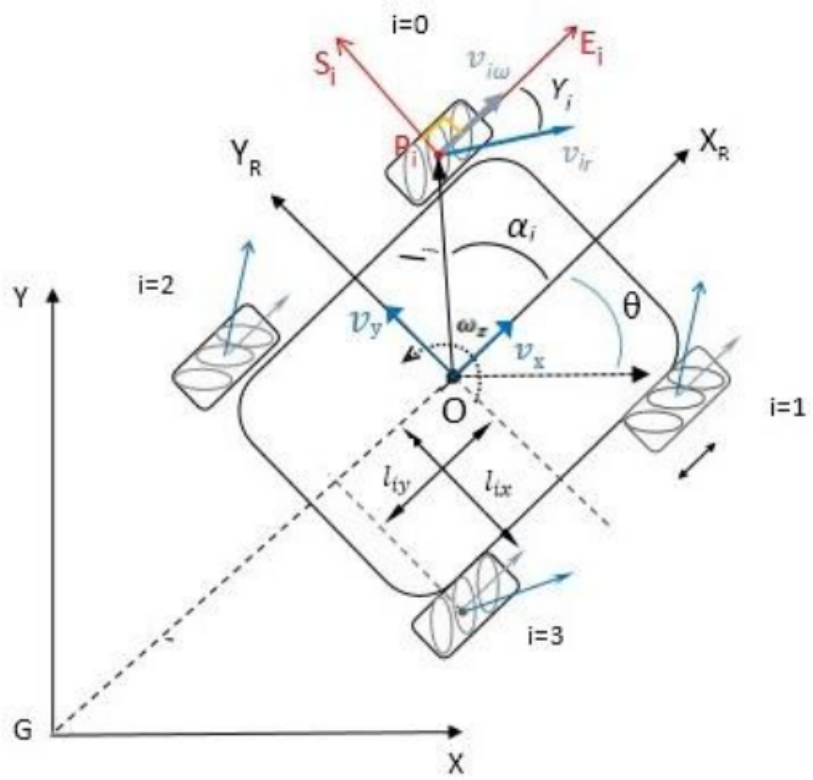
_For Autonomous

2.2 Mecanum_Wheel

2.2.1 What is Mecanum_Wheel?

The mecanum wheel is a form of tireless wheel, with a series of rubberized external rollers obliquely attached to the whole circumference of its rim. These rollers typically each have an axis of rotation at 45° to the wheel plane and at 45° to the axle line.

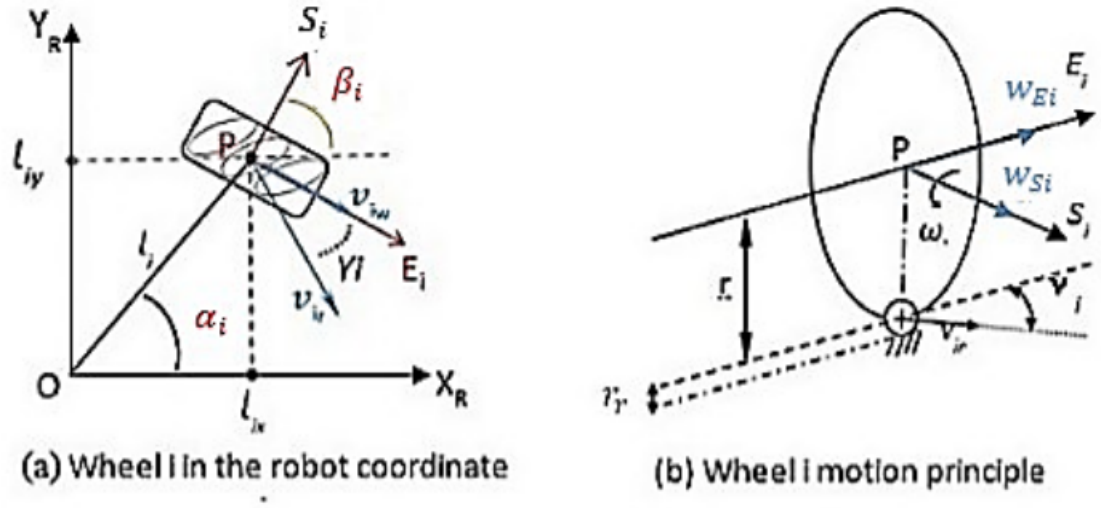
2.2.2 Kinematic of Mecanum



The configuration parameter and system velocities are defind as follows:

- x, y, θ: Robot's position (x,y) and its orientation angle θ (The angle between X and X_R);
- X,G,Y: Inertial frame;x,y are the coordinates of the reference point O in the inertial basis;
- $X_R O Y_R$: Robot's base frame; Cartesian coordinate system associated with the movement of the body center;
- S_i, P_i, E_i: Coordinate system of ith wheel in the wheel's center point P_i;
- O, P_i: The inertial basis of the Robot in Robot's frame and $P_i = \{X_{P_i}, Y_{P_i}\}$ the center of the rotation axis of the wheel i;
- $\vec{OP_i}$: is a vector that indicates the distance between Robot's center and the center of the wheel ith;
- l_{ix}, l_{iy}: l_{ix} :half of the distance between front wheels and l_{iy} : half of the distance between front wheel and the base (center of the robot O);
- l_i: distance between wheels and the base (center of the robot O);
- r_i: denotes the radius of the wheel i (Distance of the wheel's center to the roller center)
- r_r: denotes the radius of the rollers on the wheel.
- α_i: the angel between $OP_i and X_R$;
- β_i: the angle between S_i and X_R;
- γ_i: the angle between v_{ir} and E_i;
- θ[rad/s]: wheels angular velocity;
- $v_{i\theta}$[rad/s]: wheels angular velocity;
- V_{ir}: the velocity of the passive roller in the wheel i;
- $[W_{si}\ W_{Ei}\ \theta_i]^T$: Genralizd velocity of point P_i in the frame $S_i P_i E_i$;
- $[V_{si}\ V_{Ei}\ \theta_i]^T$: Generalized velocity of point P_i in the frame $X_R O Y_R$;

- $v_x v_y [m/s]$: Robot linear velocity;
- $\theta [rad/s]$:Robot angular velocity;



_Wheel i and the tangential velocity of the free roller attaced to the wheel touching the floor:

$$\mathbf{v_{ir}} = \frac{\mathbf{1}}{\mathbf{cos(45)}}\mathbf{r_r}\theta_\mathbf{i}, \quad \mathbf{W_{Ei}} = \mathbf{r_i}\theta_\mathbf{i}, \quad \mathbf{i = 0, 1, 2, 3.} \quad \textbf{(eq.1)}$$

_The velocity of the wheel i in the frame $S_i P_i E_i$, can be derived by:

$$\mathbf{v_{si}} = \mathbf{v_{ir}}\mathbf{sin}(\gamma_\mathbf{i})$$

$$\mathbf{v_{Ei}} = \theta_\mathbf{i}\mathbf{r_i} + \mathbf{v_{ir}}\mathbf{cos}(\gamma_\mathbf{i}).$$

$$\begin{bmatrix} \mathbf{v_{s_i}} \\ \mathbf{v_{E_i}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{sin}(\gamma_\mathbf{i}) \\ \mathbf{r_i} & \mathbf{cos}(\gamma_\mathbf{i}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta_i} \\ \mathbf{v_{ir}} \end{bmatrix} = \mathbf{T}_{\mathbf{P_i}}^{\mathbf{w_i}} \begin{bmatrix} \boldsymbol{\theta_i} \\ \mathbf{v_{ir}} \end{bmatrix}. \quad \textbf{(eq.2)}$$

_The transformation matrix from velocities of the ith wheel to its center:

$$T_{P_i}^{w_i} = \begin{bmatrix} \boldsymbol{\theta_i} \\ \mathbf{v_{ir}} \end{bmatrix}$$

_The velocity of the wheel's center translated to the $X_R O Y_R$coordinate system can be achieved by equation 7:

$$\begin{bmatrix} \mathbf{v_{iX_R}} \\ \mathbf{v_{iY_R}} \end{bmatrix} = \begin{bmatrix} \mathbf{cos}(\beta_\mathbf{i}) & -\mathbf{sin}(\beta_\mathbf{i}) \\ \mathbf{sin}(\beta_\mathbf{i}) & \mathbf{cos}(\beta_\mathbf{i}) \end{bmatrix} \begin{bmatrix} \mathbf{v_{S_i}} \\ \mathbf{v_{E_i}} \end{bmatrix} = T_{P_i}^{w_i} T_R^{P_i} \begin{bmatrix} \boldsymbol{\theta_i} \\ \mathbf{v_{ir}} \end{bmatrix} \quad \textbf{(eq.3)}$$

_The transformation matrix from the ith wheel's center to the robot coordinate's system can be obtained from equation

$$T_R^{P_i} = \begin{bmatrix} \mathbf{cos}(\beta_\mathbf{i}) & -\mathbf{sin}(\beta_\mathbf{i}) \\ \mathbf{sin}(\beta_\mathbf{i}) & \mathbf{cos}(\beta_\mathbf{i}) \end{bmatrix}$$

_The robot's motion is planar,we also have:

$$\begin{bmatrix} \mathbf{v_{iX_R}} \\ \mathbf{v_{iY_R}} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & -\mathbf{l_{iy}} \\ \mathbf{0} & \mathbf{1} & \mathbf{l_{ix}} \end{bmatrix} \begin{bmatrix} \mathbf{v_{X_R}} \\ \mathbf{v_{y_R}} \\ \theta_\mathbf{R} \end{bmatrix} = T' \begin{bmatrix} \mathbf{v_{X_R}} \\ \mathbf{v_{y_R}} \\ \theta_\mathbf{R} \end{bmatrix} \quad \textbf{(eq.4)}$$

Where:$T' = \begin{bmatrix} \mathbf{1} & \mathbf{0} & -\mathbf{l_{iy}} \\ \mathbf{0} & \mathbf{1} & \mathbf{l_{ix}} \end{bmatrix}$

_The inverse kinematic model can be obtained:

$$\mathbf{T}_{\mathbf{P_i}}^{\mathbf{w_i}} \mathbf{T}_{\mathbf{R}}^{\mathbf{P_i}} \begin{bmatrix} \boldsymbol{\theta_i} \\ \mathbf{v_{ir}} \end{bmatrix} = \mathbf{T}' \begin{bmatrix} \mathbf{v_{X_R}} \\ \mathbf{v_{y_R}} \\ \theta_\mathbf{R} \end{bmatrix} \quad \textbf{(eq.5)}$$

_The robot's base velocity (at point O) related to the rotational velocity of the ith wheel can be obtained from eq.6.

$$\begin{bmatrix} \boldsymbol{\theta_i} \\ \mathbf{v_{ir}} \end{bmatrix} = \mathbf{T}_{\mathbf{P_i}}^{\mathbf{w_i}} . \mathbf{T}_{\mathbf{R}}^{\mathbf{P_i}} . \mathbf{T}' \begin{bmatrix} \mathbf{v_{X_R}} \\ \mathbf{v_{y_R}} \\ \theta_{\mathbf{R}} \end{bmatrix} \quad \textbf{(eq.6)}$$

Where: $\mathrm{T}=(T_{P_i}^{w_i})^{-1}.(T_R^{P_i})^{-1}.T'$

$$\mathbf{T}=\begin{bmatrix} \mathbf{cos}(\beta_\mathbf{i}) & -\mathbf{sin}(\beta_\mathbf{i}) \\ \mathbf{sin}(\beta_\mathbf{i}) & \mathbf{cos}(\beta_\mathbf{i}) \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{0} & \mathbf{sin}(\gamma_\mathbf{i}) \\ \mathbf{r_i} & \mathbf{cos}(\gamma_\mathbf{i}) \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{1} & \mathbf{0} & -\mathbf{l_{iy}} \\ \mathbf{0} & \mathbf{1} & \mathbf{l_{ix}} \end{bmatrix}$$

Where: $l_{ix} = l_i cos(a_i)$ **and** $l_{iy} = l_i sin(a_i)$

$$\mathbf{T} = \frac{\mathbf{1}}{-\mathbf{r}} \begin{bmatrix} \frac{\mathbf{cos}(\boldsymbol{\beta}_\mathbf{i}-\gamma_\mathbf{i})}{\mathbf{sin}(\gamma_\mathbf{i})} & \frac{\mathbf{sin}(\beta\mathbf{i}-\gamma\mathbf{i})}{\mathbf{sin}(\gamma_\mathbf{i})} & \frac{\mathbf{l_i sin}(-\alpha_\mathbf{i}+\beta_\mathbf{i}-\gamma_\mathbf{i})}{\mathbf{sin}(\gamma_\mathbf{i}} \\ -\frac{\mathbf{rcos}(\beta_\mathbf{i})}{\mathbf{sin}(\gamma_\mathbf{i}} & -\frac{\mathbf{rcos}(\beta\mathbf{i})}{\mathbf{sin}(\gamma_\mathbf{i})} & -\frac{\mathbf{l_i sin}(-\alpha_\mathbf{i}+\beta_\mathbf{i})\mathbf{r}}{\mathbf{sin}(\gamma_\mathbf{i})} \end{bmatrix} \quad \textbf{(eq.7)}$$

Since there is a relation between independent variables v_{ir} and ω_i in each joint and the systems angular and linear velocity, assuming that there is no wheel slipping on the ground, the system inverse kinematic can be obtained by eq.8.

_How to calculate Velocity of 4_wheel:

$$\begin{bmatrix} \boldsymbol{\omega_1} \\ \boldsymbol{\omega_2} \\ \boldsymbol{\omega_3} \\ \boldsymbol{\omega_4} \end{bmatrix} = \frac{-\mathbf{1}}{\mathbf{r}} \begin{bmatrix} \frac{\mathbf{cos}(\boldsymbol{\beta_1}-\gamma_1)}{\mathbf{sin}(\gamma_1)} & \frac{\mathbf{sin}(\beta_1-\gamma_1)}{\mathbf{sin}(\gamma_1)} & \frac{\mathbf{l_1 sin}(-\alpha_1+\beta_1-\gamma_1)}{\mathbf{sin}(\gamma_\mathbf{i}} \\ \frac{\mathbf{cos}(\beta_2-\gamma_2)}{\mathbf{sin}(\gamma_2)} & \frac{\mathbf{sin}(\beta\mathbf{2}-\gamma\mathbf{2})}{\mathbf{sin}(\gamma_2)} & \frac{\mathbf{l_2 sin}(-\alpha_2+\beta_2-\gamma_2)}{\mathbf{sin}(\gamma_2} \\ \frac{\mathbf{cos}(\beta_3-\gamma_3)}{\mathbf{sin}(\gamma_3)} & \frac{\mathbf{sin}(\beta\mathbf{3}-\gamma_3)}{\mathbf{sin}(\gamma_3)} & \frac{\mathbf{l_3 sin}(-\alpha_3+\beta_3-\gamma_\mathbf{i})}{\mathbf{sin}(\gamma_3} \\ \frac{\mathbf{cos}(\beta_4-\gamma_4)}{\mathbf{sin}(\gamma_4)} & \frac{\mathbf{sin}(\beta\mathbf{4}-\gamma\mathbf{4})}{\mathbf{sin}(\gamma_4)} & \frac{\mathbf{l_4 sin}(-\alpha_4+\beta_4-\gamma_4)}{\mathbf{sin}(\gamma_4} \end{bmatrix} \begin{bmatrix} \mathbf{V_x} \\ \mathbf{V_y} \\ \boldsymbol{\omega_z} \end{bmatrix} \quad \textbf{(eq.8)}$$

2.2.3 FOUR MECANUM OMNIDIRECTIONAL SOLUTION

Table 1. Robot Parameters

i	Wheels	α_i	β_i	γ_i	l_i	l_{ix}	l_{iy}
0	1sw	$\pi/4$	$\pi/2$	$-\pi/4$	l	l_x	l_y
1	2sw	$-\pi/4$	$-\pi/2$	$\pi/4$	l	l_x	l_y
2	3sw	$3\pi/4$	$\pi/2$	$\pi/4$	l	l_x	l_y
3	4sw	$-3\pi/4$	$-\pi/2$	$-\pi/4$	l	l_x	l_y

_We get Inverse kinematic form:

$$\begin{bmatrix} \boldsymbol{\omega_1} \\ \boldsymbol{\omega_2} \\ \boldsymbol{\omega_3} \\ \boldsymbol{\omega_4} \end{bmatrix} = \frac{\mathbf{1}}{\mathbf{r}} \begin{bmatrix} \mathbf{1} & \mathbf{-1} & \mathbf{-(l_x + l_y)} \\ \mathbf{1} & \mathbf{1} & \mathbf{(l_x + l_y)} \\ \mathbf{1} & \mathbf{1} & \mathbf{-(l_x + l_y)} \\ \mathbf{1} & \mathbf{-1} & \mathbf{(l_x + l_y)} \end{bmatrix} \begin{bmatrix} \mathbf{V_x} \\ \mathbf{V_y} \\ \boldsymbol{\omega_z} \end{bmatrix} \quad \textbf{(eq.9)}$$

_And Forward form:

$$\begin{bmatrix} \mathbf{V_x} \\ \mathbf{V_y} \\ \boldsymbol{\omega_z} \end{bmatrix} = \frac{\mathbf{r}}{\mathbf{4}} \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{-1} & \mathbf{1} & \mathbf{1} & \mathbf{-1} \\ -\frac{\mathbf{1}}{\mathbf{(l_x+l_y)}} & \frac{\mathbf{1}}{\mathbf{(l_x+l_y)}} & -\frac{\mathbf{1}}{\mathbf{(l_x+l_y)}} & \frac{\mathbf{1}}{\mathbf{(l_x+l_y)}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega_1} \\ \boldsymbol{\omega_2} \\ \boldsymbol{\omega_3} \\ \boldsymbol{\omega_4} \end{bmatrix} \quad \textbf{(eq.10)}$$

_The resultant velocity and its direction in the stationery coordinate axis (x, y, z) can be achieved by the following equations (eq.11,12):

$$\boldsymbol{\rho}= \mathbf{tan}^{-1}(\frac{\mathbf{V_y}}{\mathbf{V_x}}), \quad \textbf{(eq.11)}$$

$$V_R = \sqrt{V_x^2 + V_y^2} \quad \textbf{(eq.12)}$$

2.2.4 Kinematic of Wheel for write Program

- **Forward Kinematic**

$$V_x = \frac{R}{4} * (\omega_1 + \omega_2 + \omega_3 + \omega_4)$$

$$V_y = \frac{R}{4} * (-\omega_1 + \omega_2 + \omega_3 - \omega_4)$$

$$\omega_z = \frac{R}{4(l_x + l_y)} * (-\omega_1 + \omega_2 - \omega_3 + \omega_4)$$

- **Inverse Kinematic**

$$\omega_1 = \frac{4}{R} * (V_x - V_y - \omega_z(l_x + l_y))$$

$$\omega_2 = \frac{4}{R} * (V_x + V_y + \omega_z(l_x + l_y))$$

$$\omega_3 = \frac{4}{R} * (V_x + V_y - \omega_z(l_x + l_y))$$

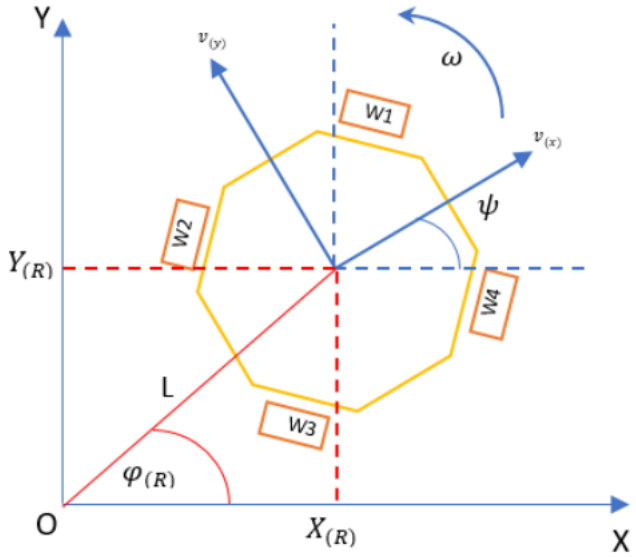
$$\omega_4 = \frac{4}{R} * (V_x - V_y + \omega_z(l_x + l_y))$$

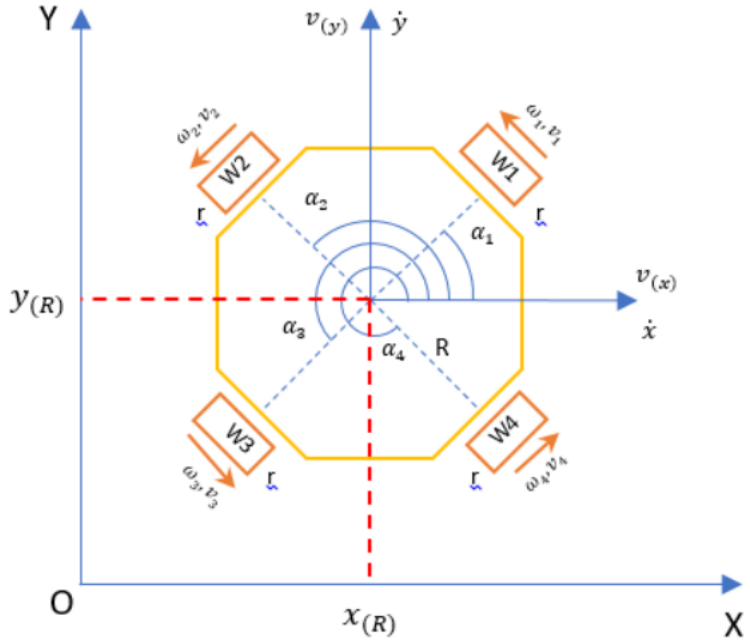
2.3 OMNI_WHEEL

2.3.1 What is Omni_Wheel?

Omni wheels or poly wheels, similar to Mecanum wheels, are wheels with small discs (called rollers) around the circumference which are perpendicular to the turning direction. The effect is that the wheel can be driven with full force, but will also slide laterally with great ease.

2.3.2 Kinematic Modeling of Omni





The configuration parameter:

- $(\boldsymbol{\alpha_1, \alpha_2, \alpha_3, \alpha_4})$ = The angle formed between the wheels and the robot reference point .
- $(\boldsymbol{\omega_w} =)$ The angular velocity of each wheel.
- $\mathbf{V_w}$ = The linear velocity of each Wheel.
- **XOY**= global coordinates are notified using $\mathbf{V_{(g)}} = [\dot{\mathbf{X}}\ \ \dot{\mathbf{Y}}\ \ \boldsymbol{\omega}]$.
- **R**= is the distance from the Wheel to the center of the robot.
- **r**= is the radius of the omni wheel.
- $\mathbf{V_R} = [\mathbf{V_{(x}}\ \ \mathbf{V_y}]$ = The position of the robot's coordinate.
- $\mathbf{X_{(g)}} = [\mathbf{X_R}\ \ \mathbf{Y_R}\ \ \boldsymbol{\Psi}]^{\mathbf{T}}$ = Orientation to global coordinates.
- $\boldsymbol{\omega}$ = is the notation of the angular velocity of the robot to the global coordinates.
- Ψ_R= is the direction of movement of the robot at global coordinates.
- Ψ_n= is a notation of the direction of movement of the robot to global coordinates.
- **L**= is the notation of the resultant linear velocity of the robot.

Formulate the following equations:

$$\boldsymbol{\psi_{(R)}} = \mathbf{tan^{-1}}\frac{\mathbf{Y_R}}{\mathbf{X_R}} \quad \textbf{(eq.1)}$$

$$\mathbf{L} = \sqrt{\mathbf{X^2_{(R}} + \mathbf{Y^2_R}} \quad \textbf{(eq.2)}$$

$$\begin{bmatrix} \mathbf{V_x} \\ \mathbf{V_y} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{cos}(\boldsymbol{\psi}) & \mathbf{sin}(\psi) & \mathbf{0} \\ -\mathbf{sin}(\psi & \mathbf{cos}(\boldsymbol{\psi}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} . \begin{bmatrix} \dot{\mathbf{X}} \\ \dot{\mathbf{Y}} \\ \boldsymbol{\omega} \end{bmatrix} \quad \textbf{(eq.3)}$$

INVERSE KINEMATIC:

$$\mathbf{V_w} \begin{bmatrix} \boldsymbol{\omega_1} \\ \boldsymbol{\omega_2} \\ \boldsymbol{\omega_3} \\ \boldsymbol{\omega_4} \end{bmatrix} = \begin{bmatrix} -\mathbf{sin}(\psi + \alpha_1) & \mathbf{cos}(\psi + \alpha_1) & R \\ -\mathbf{sin}(\psi + \alpha_2) & \mathbf{cos}(\psi + \alpha_2) & R \\ -\mathbf{sin}(\psi + \alpha_3) & \mathbf{cos}(\psi + \alpha_3) & R \\ -\mathbf{sin}(\psi + \alpha_4) & \mathbf{cos}(\psi + \alpha_4) & R \end{bmatrix} \begin{bmatrix} \mathbf{V_x} \\ \mathbf{V_y} \\ \boldsymbol{\omega} \end{bmatrix} \quad \textbf{(eq.4)}$$

Where $\boldsymbol{\alpha_1} = \frac{\pi}{4}, \boldsymbol{\alpha_2} = \frac{3\pi}{4}, \boldsymbol{\alpha_3} = \frac{5\pi}{4}, \boldsymbol{\alpha_4} = \frac{7\pi}{4}$.
So we get:

$$\mathbf{V_w}\begin{bmatrix}\boldsymbol{\omega_1}\\ \boldsymbol{\omega_2}\\ \boldsymbol{\omega_3}\\ \boldsymbol{\omega_4}\end{bmatrix} = \begin{bmatrix}-\mathbf{sin}(\psi+\frac{\pi}{4}) & \mathbf{cos}(\psi+\frac{\pi}{4}) & R\\ -\mathbf{sin}(\psi+\frac{3\pi}{4}) & \mathbf{cos}(\psi+\frac{3\pi}{4}) & R\\ -\mathbf{sin}(\psi+\frac{5\pi}{4}) & \mathbf{cos}(\psi+\frac{5\pi}{4}) & R\\ -\mathbf{sin}(\psi+\frac{7\pi}{4}) & \mathbf{cos}(\psi+\frac{7\pi}{4}) & R\end{bmatrix}\begin{bmatrix}\mathbf{V_x}\\ \mathbf{V_y}\\ \boldsymbol{\omega}\end{bmatrix} \quad \textbf{(eq.5)}$$

FORWARD KINEMATIC

$$\mathbf{V_g} = \begin{bmatrix}\mathbf{V_x}\\ \mathbf{V_y}\\ \boldsymbol{\omega}\end{bmatrix} = \frac{\mathbf{1}}{\mathbf{2}}\begin{bmatrix}-\mathbf{sin}(\psi+\frac{\pi}{4}) & -\mathbf{sin}(\psi+\frac{3\pi}{4}) & -\mathbf{sin}(\psi+\frac{5\pi}{4}) & -\mathbf{sin}(\psi+\frac{7\pi}{4})\\ \mathbf{cos}(\psi+\frac{\pi}{4}) & \mathbf{cos}(\psi+\frac{3\pi}{4}) & \mathbf{cos}(\psi+\frac{5\pi}{4}) & \mathbf{cos}(\psi+\frac{7\pi}{4})\\ \frac{\mathbf{1}}{\mathbf{2R}} & \frac{\mathbf{1}}{\mathbf{2R}} & \frac{\mathbf{1}}{\mathbf{2R}} & \frac{\mathbf{1}}{\mathbf{2R}}\end{bmatrix}\begin{bmatrix}\boldsymbol{\omega_1}\\ \boldsymbol{\omega_2}\\ \boldsymbol{\omega_3}\\ \boldsymbol{\omega_4}\end{bmatrix}$$

(eq.6)

2.3.3 Kinematic of Wheel for write Program

- **Forward Kinematic**
  For $\psi = 0$ because I want to take position start of robot in coordinate P(x=0,y=0,ψ = 0)

$$\boldsymbol{V_x} = \frac{\sqrt{\mathbf{2}}\boldsymbol{r}}{\mathbf{4}} * (-\boldsymbol{\omega_1} - \boldsymbol{\omega_2} + \boldsymbol{\omega_3} + \boldsymbol{\omega_4})$$

$$\boldsymbol{V_y} = \frac{\sqrt{\mathbf{2}}\boldsymbol{r}}{\mathbf{4}} * (\boldsymbol{\omega_1} - \boldsymbol{\omega_2} - \boldsymbol{\omega_3} + \boldsymbol{\omega_4})$$

$$\boldsymbol{\omega_z} = \frac{\mathbf{1}\boldsymbol{r}}{\mathbf{4}\boldsymbol{R}} * (\boldsymbol{\omega_1} + \boldsymbol{\omega_2} + \boldsymbol{\omega_3} + \boldsymbol{\omega_4})$$

- **Inverse Kinematic**

$$\boldsymbol{\omega_1} = \frac{\sqrt{\mathbf{2}}}{\mathbf{2}\boldsymbol{r}} * (-\boldsymbol{V_x} + \boldsymbol{V_y} + \boldsymbol{\omega_z} * \boldsymbol{R})$$

$$\boldsymbol{\omega_2} = \frac{\sqrt{\mathbf{2}}}{\mathbf{2}\boldsymbol{r}} * (-\boldsymbol{V_x} - \boldsymbol{V_y} + \boldsymbol{\omega_z} * \boldsymbol{R})$$

$$\boldsymbol{\omega_3} = \frac{\sqrt{\mathbf{2}}}{\mathbf{2}\boldsymbol{r}} * (\boldsymbol{V_x} + \boldsymbol{V_y} - \boldsymbol{\omega_z} * \boldsymbol{R})$$

$$\boldsymbol{\omega_4} = \frac{\sqrt{\mathbf{2}}}{\mathbf{2}\boldsymbol{r}} * (\boldsymbol{V_x} + \boldsymbol{V_y} + \boldsymbol{\omega_z} * \boldsymbol{R})$$

Where: $\boldsymbol{\omega_i}$:Angular Velocity of Wheel, $\boldsymbol{\omega_z}$:Angular velocity Z:yaw
r:radius of wheel, **R**:distance from wheel to the center robot.

2.4 SWERVE DRIVE

3. Stepper Motor

3.1 What is Stepper Motor?

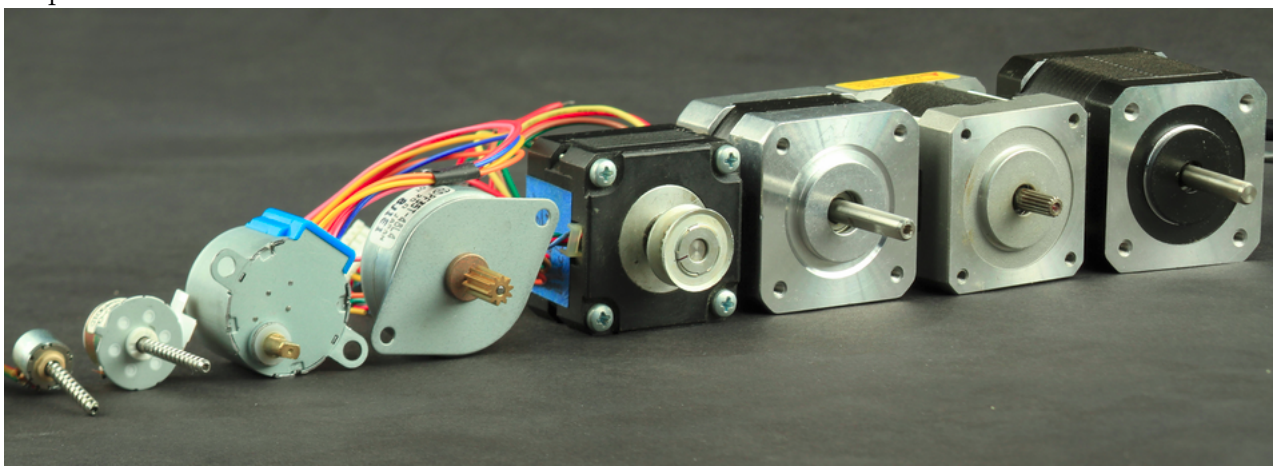
Stepper motors are DC motors that move in discrete steps. They have multiple coils that are organized in groups called "phases". By energizing each phase in sequence, the motor will rotate, one step at a time.

What are stepper good for?

A Stepper motor has maximum torque at low speeds, so they are a good choice for applications requiring low speed with high precision.

3.2 Types of Steppers

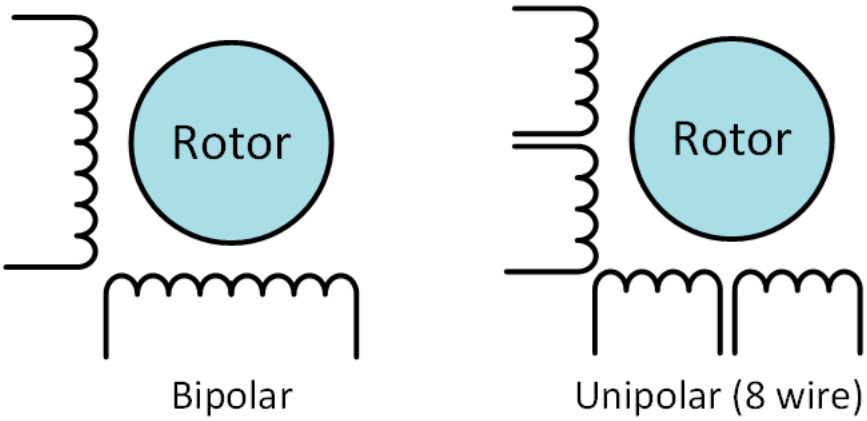
These are: Permanent Magnet or Hybrid steppers, either 2-phase bipolar, or 4-phase unipolar.



- Motor Size
  - Stepper motors come in sizes ranging from smaller than a peanut to big NEMA 57 monsters.
  - Most motors have torque ratings. This is what you need to look at to decide if the motor has the strength to do what you want.
  - NEMA 17 is a common size used in 3D printers and smaller CNC mills. Smaller motors find applications in many robotic and animatronic applications.
  - The NEMA numbers define standard faceplate dimensions for mounting the motor.
- Step Count
  - The number of steps per revolution ranges from 4 to 400.
  - Commonly available step counts are 24, 48 and 200.
  - Resolution is often expressed as degrees per step. A 1.8° motor is the same as a 200 step/revolution motor.
  - High step count motors top-out at lower RPMs than similar size.
  - The higher step-rates needed to turn these motors results in lower torque than a similar size low-step-count motor at similar speeds.
- Gearing
  - Another way to achieve high positioning resolution is with gearing. A 32:1 gear-train applied to the output of an 8-steps/revolution motor will result in a 256 step motor.
  - A gear train will also increase the torque of the motor. Some tiny geared steppers are capable of impressive torque. But the tradeoff of course is speed. Geared stepper motors are generally limited to low RPM applications.
  - Backlash is another issue with geared motors. When the motor reverses direction, it needs to take up any slack there may be in the gear-train. This can affect positioning accuracy.
- Shaft Style

  Motors are available with a number of shaft styles:
  - Round or "D" Shaft: These are available in a variety of standard diameters and there are many pulleys, gears and shaft couplers designed to fit. "D" shafts have one flattened side to help prevent slippage. These are desirable when high torques are involved.

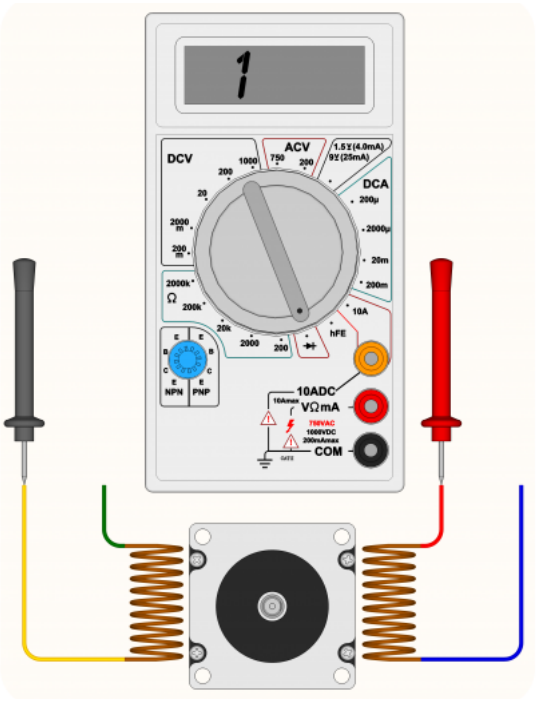
  - Geared shaft: Some shafts have gear teeth milled right into them. These are typically designed to mate with modular gear trains.
  - Lead-Screw Shaft: Motors with lead-screw shafts are used to build linear actuators. Miniature versions of these can be found as head positioners in many disk drives.
- Wiring
  These are Permanent Magnet or Hybrid steppers wired as 2-phase bipolar, or 4-phase unipolar.
- Coils and Phases
  - A stepper motor may have any number of coils.
  - These are connected in groups called "phases".



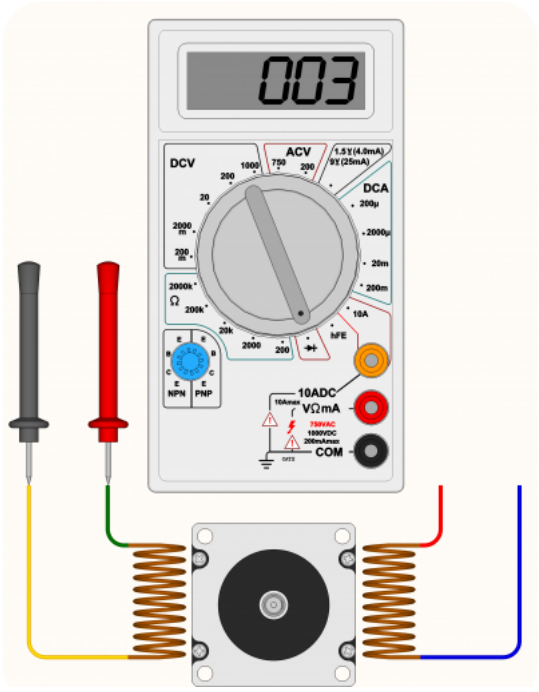
- Unipolar vs. Bipolar
  - Unipolar drivers, always energize the phases in the same way. One lead, the "common" lead, will always be negative. The other lead will always be positive. Unipolar drivers can be implemented with simple transistor circuitry. The disadvantage is that there is less available torque because only half of the coils can be energized at a time.
  - Bipolar drivers use H-bridge circuitry to actually reverse the current flow through the phases. By energizing the phases with alternating the polarity, all the coils can be put to work turning the motor.

3.3 How to identify the motor coil wiring pairs

- Set the Multimeter to the diode test position in
- The Multimeter will not make any sound, and the display will still be "1" so the coil not pair.



- when The Multimeter may sound a soft squeak, and a different number will appear on the display instead of the digit one.



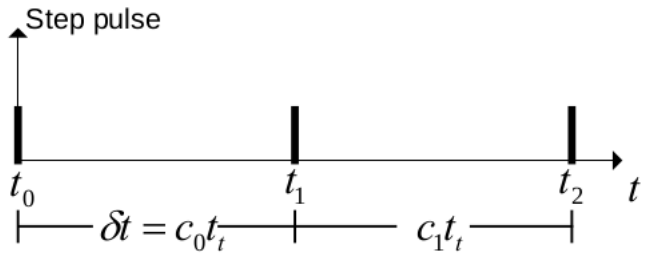
Separate them from the others and connect to the connectors of the stepper driver A+/A- and B+/B-.

3.4 How to Control Stepper_Motor

3.4.1 Fundamental stepper motor equations

To rotate the stepper motor at a constant speed, pulses must be generated at a steady rate, shown in Figure:

Figure 2-4. Stepper motor pulses



- A counter generates these pulses, running at the frequency **f t** [**Hz**].
- The delay t programmed by the counter c is $\boldsymbol{\delta t = tC = \frac{C}{f}}$.
- The motor step angle α , position θ , and speed ω are given by :

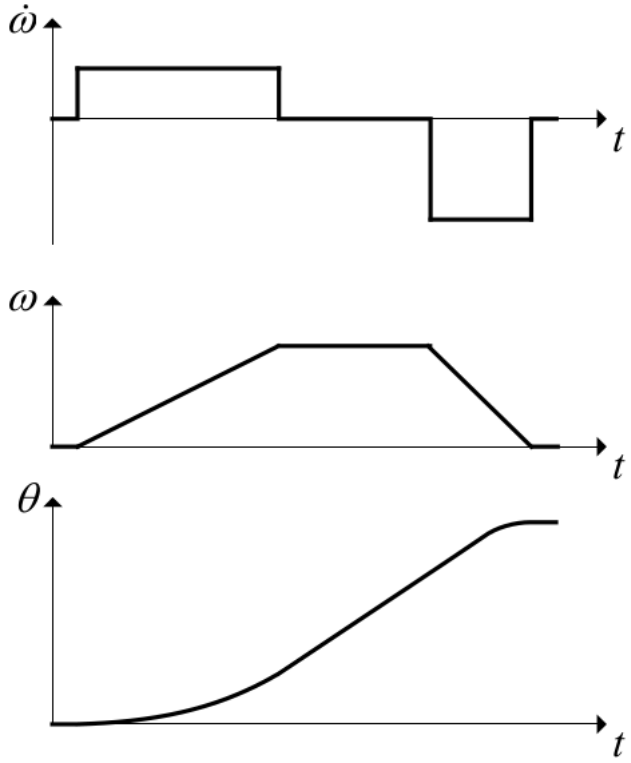
$$\boldsymbol{\alpha = \frac{2\pi}{spr}[rad], \quad \theta = n\alpha[rad], \quad \omega = \frac{\alpha}{\delta t}[rd/s]}$$

where **spr** is the number of steps per round, **n** is the number of steps, and 1 rad/sec = 9,55 rpm.

3.4.2 **Linear speed ramp**

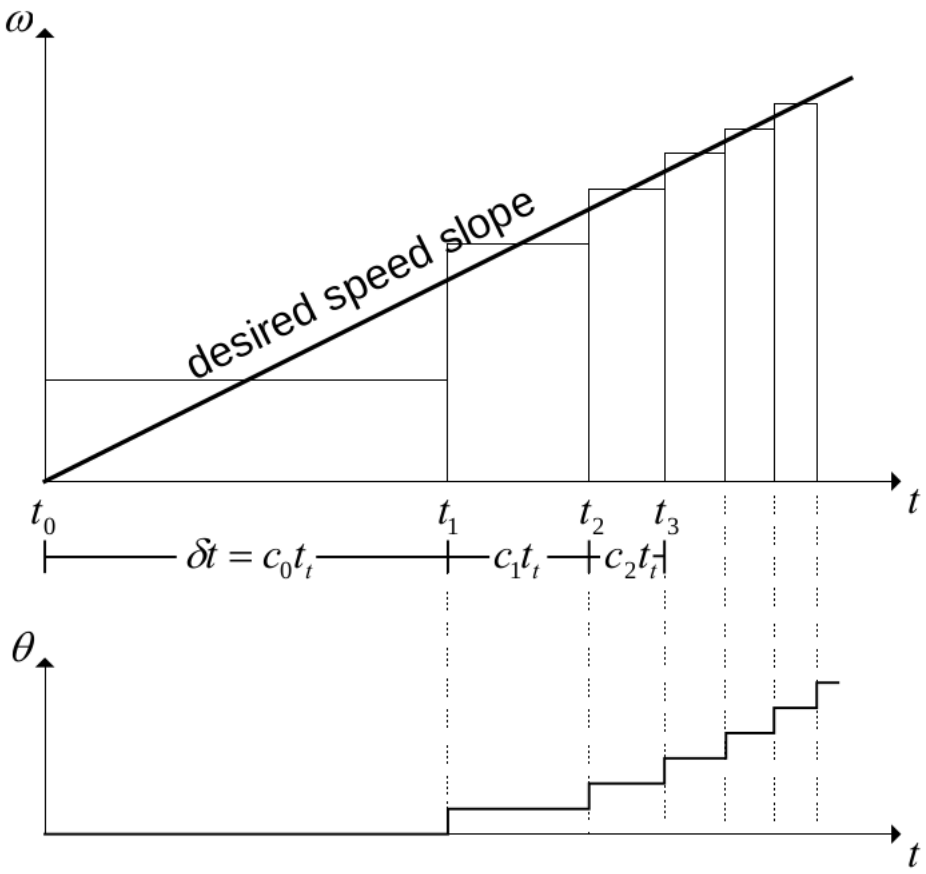
To start and stop the stepper motor in a smooth way, control of the acceleration and deceleration is needed. Figure shows the relation between acceleration, speed and position. Using a constant acceleration/deceleration gives a linear speed profile.

Figure: Acceleration (&) , speed ( ) and position ( ).



- The time delay δt between the stepper motor pulses controls the speed.
- These time delays t must be calculated in order to make the speed of the stepper motor follow the speed ramp as closely as possible.
- Discrete steps control the stepper motor motion, and the resolution of the time delay between these steps is given by the frequency of the timer.

Figure. Speed profile vs. stepper motor pulses/speed.



3.4.3 Exact calculations of the inter-step delay

The first counter delay c_0 as well as succeeding counter delays c n , are given by (see appendix for details):

$$\boldsymbol{C_0 = \frac{1}{t_t}\sqrt{\frac{2\alpha}{\dot{\omega}}}, \quad C_n = C_0(\sqrt{n+1} - \sqrt{n}}$$

The counter value at the time n , using Taylor series approximation for the inter-step delay (see appendix for details) is given by:

$$\boldsymbol{C_n = C_{n-1} - \frac{2C_{n-1}}{4n+1}}$$

- The acceleration is given by c 0 and n . If a change in acceleration (or deceleration) is done, a new n must be calculated.
- The time t n and n as a function of the motor acceleration, speed and step angle are given by

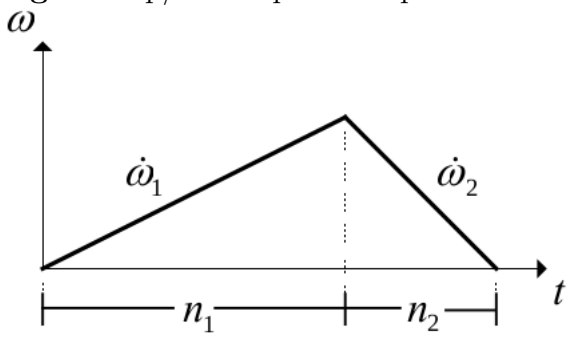
$$\boldsymbol{t_n = \frac{\omega_n}{\dot{\omega}}}$$

- Merging these equation gives the relationship

$$\boldsymbol{n\dot{\omega} = \frac{\omega^2}{2\alpha}}$$

- This shows that the number of steps needed to reach a given speed is inversely proportional to the acceleration: $\boldsymbol{n_1\dot{\omega}_1 = n_2\dot{\omega}_2}$

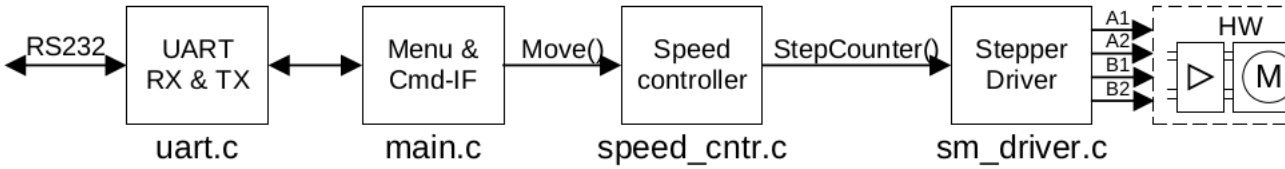
Figure. Up/down speed ramp



3.5 Implementation

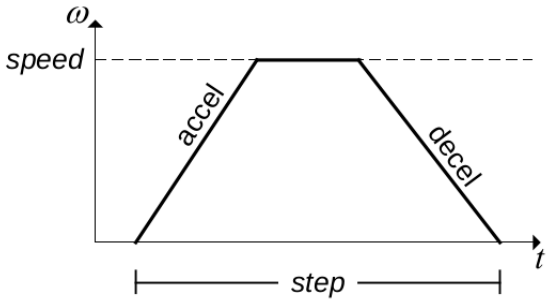
The demo application is divided in three major blocks, as shown in the block diagram in Figure. There is one file for each block and also a file for UART routines used by the main routine.

Figure 3-1. Block diagram of demo application



- main.c has a menu and a command interface, giving the user control of the stepper motor by a terminal connected to the serial line.
- speed_cntr.c calculates the needed data and generates step pulses to make the stepper motor follow the desired speed profile.
- sm_driver.c counts the steps and outputs the correct signals to control the stepper motor.

Figure Speed profile



The parameters describing the speed profile is:

- step - Number of steps to move.
- accel - Acceleration to use.
- decel - Deceleration to use.
- speed - (Maximum) speed to use.

3.5.1 Menu and command interface

The UART setting is 19200 baud, 8 data bit, none parity and 1 stop bits. Any terminal emulation program should work. Using the terminal the user can give different commands to control the stepper motor and get information back from the demo application.

The UART RX interrupt routine (found in uart.c) stores received characters in the receiver buffer and handles backspace. When ¡enter¿ (ascii code 13) is received the main routine reads the buffer and executes the given command.

- When starting, and on the '?' command, this help screen is shown:

```
-----------------------------------------------------------------
Atmel AVR446 - Linear speed control of stepper motor

?        - Show help
a [data] - Set acceleration (range: 71 - 32000)
d [data] - Set deceleration (range: 71 - 32000)
s [data] - Set speed (range: 12 - motor limit)
m [data] - Move [data] steps (range: -64000 - 64000)
move [steps] [accel] [decel] [speed]
         - Move with all parameters given
<enter>  - Repeat last move

    acc/dec data given in 0.01*rad/sec^2 (100 = 1 rad/sec^2)
    speed data given in 0.01*rad/sec (100 = 1 rad/sec)
-----------------------------------------------------------------
```

- After the menu is shown or a command is executed the info line is shown:

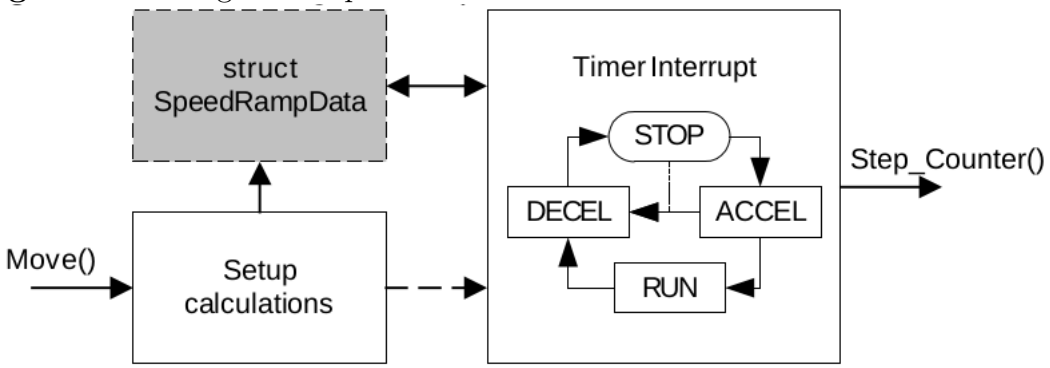
  Motor pos: 0 , a:4000, d:4000, s:2000, m:400

- The demo application gives the current motor position, acceleration, deceleration, and speed settings, as well as the number of steps to move.

3.5.2 Speed controller

The speed controller calculates and generates the speed profile.To run the stepper motor, the speed controller is set up by calling the function Move().

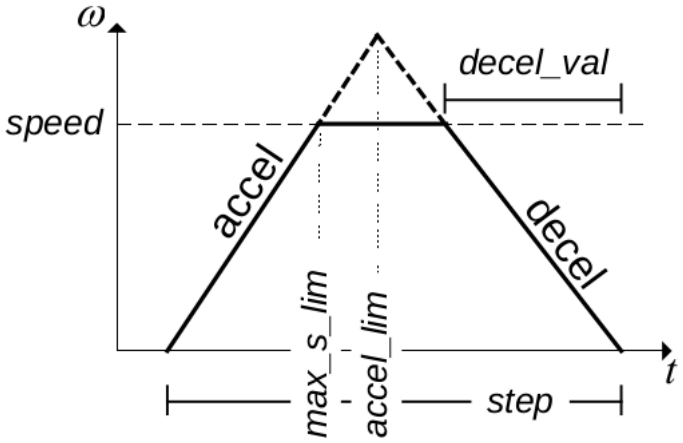
Figure Block diagram of speed controller



3.5.3 Setup calculations

- Find speed:
  - A_T_x100 = f_t*100
  - min_delay=C=$\frac{A\_T\_x100}{speed}$
- Find acceleration:
  - T1_ FREQ_148 =0.676f_t/ 100
  - A_SQ = 2α10000000000
  - step_delay=$C_0 = T1\_FREQ\_148\sqrt{\frac{A\_SQ}{accel}}/100$
- Acceleration continues until desired speed is reached

Figure Speed ramp limited by desired speed value.

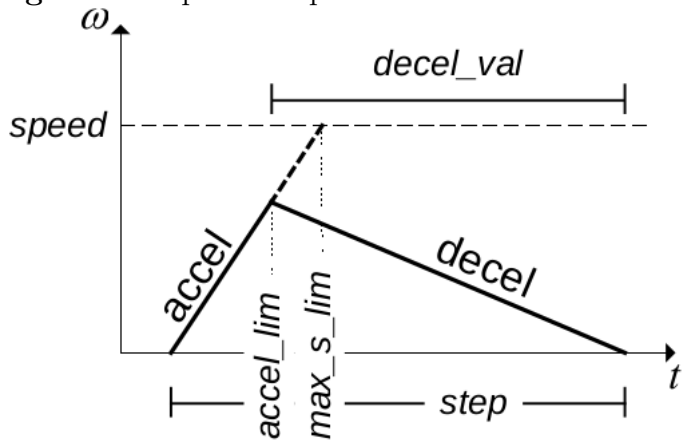


– max_s_lim is the number of steps needed to accelerate to the desired speed. max_s_lim=n=$\frac{speed^2}{2\alpha * accel * 100}$
– accel_lim is the number of steps before deceleration starts (disregarding desired speed).
accel_lim=$n_1 = \frac{step * decel}{accel + decel}$

Ifmax_s_lim ¡ accel_lim the acceleration is limited by reaching desired speed. The deceleration depends on this, and in this case decal_val is found by: decel_val=-max_s_lim*$\frac{accel}{decel}$

- Deceleration starts before desired speed is reached

Figure 3-5 Speed ramp with deceleration start before desired speed reached



If max_s_lim ¿ accel_lim the acceleration is limited by deceleration start, decal_val is then found by:
$decal_val = -(step - accel_lim)$

3.5.4 Timer interrupt

The timer interrupt generates the 'step pulse'(calls the function StepCounter()) and is only runnig when the stepper motor is moving.The timer interrupt will operate in four different states according to the speed profile,

Figure3-6: Operating state for different speed profile parts

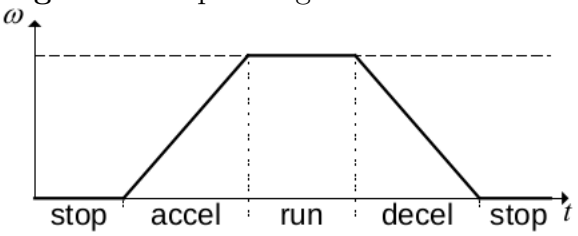
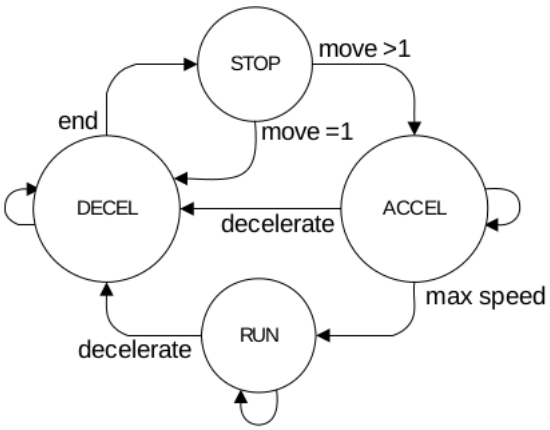


Figure3-7: State machine for timer interrupt



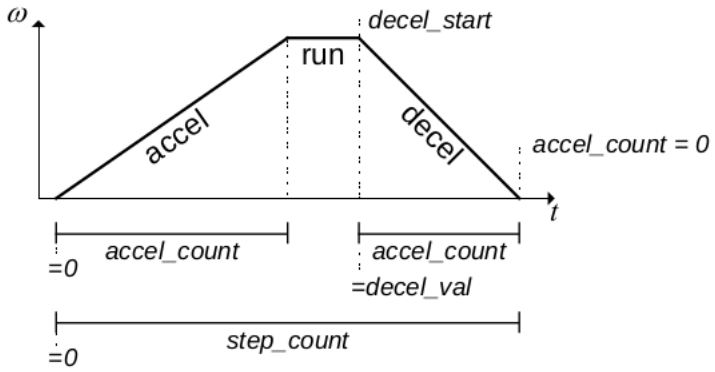
Calculations and counters

For each step during acceleration and deceleration a new time delay must be calculated. This calculation includes a division giving a remainder, and to improve accuracy this remainder is kept and included in the next calculation.

$$\boldsymbol{new\_step\_delay = step\_delay - \frac{2*step\_delay+rest}{4*accel\_count+1}}$$

$$\boldsymbol{new_rest = (2 * step\_delay + rest)(mod(4 * accel\_count + 1))}$$

Figure3-8: Counting variables in time delay



3.6 Code size and speed

The timer interrupt performs calculations during acceleration and deceleration and approximately 200us are used in one timer interrupt. When running at constant speed less time is need and approximately 35us is sufficient. The maximum output speed is then limited by acceleration/deceleration calculations. For a stepper motor with 400 steps per round maximum output speed is:

$$\boldsymbol{\omega_{max} = \frac{2\pi}{200us * 400} = 78,5rad/sec(= 750rpm)}$$

where Rpm=(rad/s)*(2/π)

4. Servo Motor

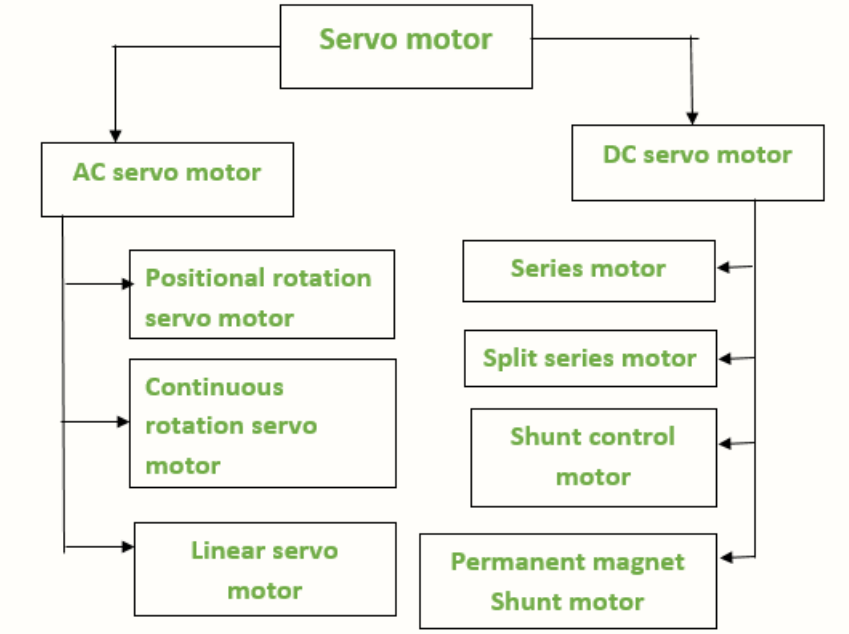
4.1 What is Servo motor?

A servomotor is a packaged of several components: a motor (usually electric, although fluid power motors may also be used), a gear train to reduce the many rotations of the motor to a higher torque rotation, a position encoder that identifies the position of the output shaft and an inbuilt control system.The working principle of the servo motor and electromagnetic motor are same except the structure and the function are dissimilar. The standard servo motor uses the plastic gear whereas the high power servo motor uses the metal gear.

They are suitable for many applications due to the progress of the microprocessor, power transistor high precision control. they include three wires power, ground, and

control. Based on the size and outline, these motors are used in various applications. The most frequently used servo motor is RC servo motor which is mostly used in hobby applications. Mainly, this motor includes affordability, simplicity, and consistency.

4.2 **What are the Types of Servo Motor?** There are two main types of this motor based on the supply used for its function. They are AC servo motors & DC servo motors. Check the below flow chart for the types of servo motor.



4.1.1 AC Servo Motor

An ac servo motor includes an encoder. The controller uses this encoder to give the feedback as well as closed loop control. To achieve more torque, theses motors have superior designs. Mostly, an AC servo motor uses in robotics, automation, CNC equipment, and many more applications.

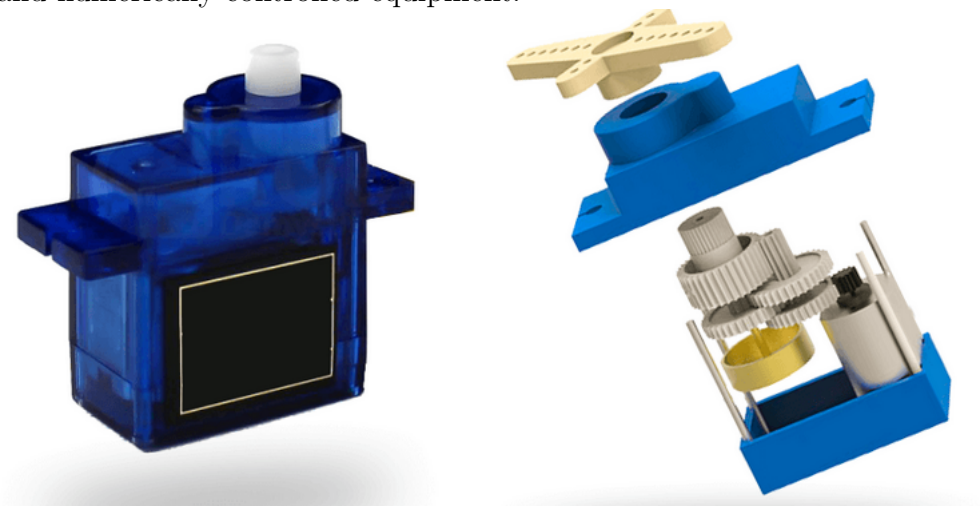
- **Positional rotation servo motor**
  In positional rotation motor, the output of the shaft in motor rotates with 180 degrees. This type of motor mainly comprises includes physical stops that places in the gear mechanism to prevent rotating outside to protect the rotation sensor. The positional rotation servo motor uses in robots, aircraft, toys, controlled cars, / many more applications.
- **Continuous rotation servo motor**
  Continuous rotation servo motor is quite related to the common positional rotation servo motor. But it can go in any direction indefinitely. The control signal, rather than set the static position of the servo, is understood as the speed and direction of rotation. The range of potential commands sources the servo to rotate clockwise or anticlockwise. Also, it will change the speed, depending on the command signal. This type of motor uses in a radar dish if you are riding

one on a robot or you can use one as a drive motor on a mobile robot.

- **Linear Servo motor**
  Linear servo motor is similar to positional rotation servo motor. It has an extra gears and with an extra gears, it alters the o/p from circular to back-and-forth. These servo motors are not simple to find, but sometimes you can find them at hobby stores where they use as actuators in higher model airplanes.

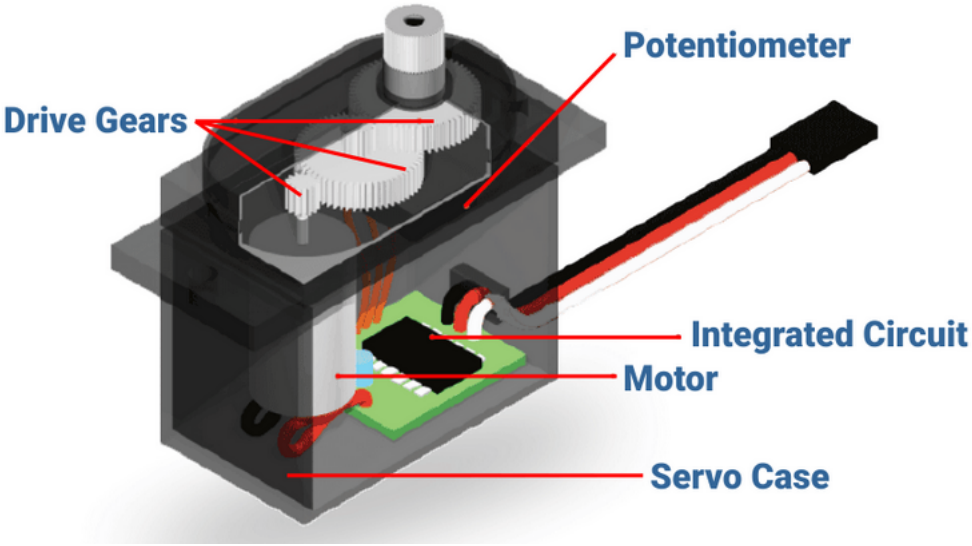
4.1.2 DC Servo Motor

The DC servo motor includes a DC source separately in the field of the armature winding. The motor can be controlled by managing the field current or the armature current. Both the armature control and field control have benefits. This motor offers a quick and accurate response to begin or end command signals because of the small armature inductive reactance. They utilize in several devices and numerically controlled equipment.



- **Series servo motor**
  The series servo motors include high starting torque as well as draws huge current. The speed regulation of this motor is very less. Turnaround can be attained by overturning the field voltage polarity using split series field winding.
- **Split series servo motor**
  A split series motors can function as an individually energized field controlled motor. The motor armature supplies with a stable current supply. This motor has a common curve with torque speed. This specifies high stall torque & a fast decline in torque by amplifying in speed.
- **Shunt Control Motor**
  The shunt control motor includes two windings such as field windings and armature windings. Field windings are located on the stator whereas the armature windings are located on the rotor of machine. In a DC shunt motor, the two windings connect across the DC source in parallel.
- **Permanent Magnet Shunt Motor**
  It is a permanent excitation motor wherever a stable magnet supplies the field. The motor performance is same to armature controlled permanent field motor.

4.3 Construction of Servo Motor



- The DC motor connects with a gear mechanism which provides feedback to a position sensor which is mostly a potentiometer.
- The standard servo motor uses the plastic gear whereas the high power servo motor uses the metal gear.
- The gear box or gear reduction unit, the output of the motor delivers via servo spline to the servo arm. the gear box is formed by gears which may increase or decrease the speed and torque.
- A control circuit allows for control over the motor's motion by sending electric pulses Motor consists of three wires- a black wire connected to ground. A white/yellow wire connected to control unit. And a red wire connected to power supply.

4.4 Advantages of Servo Motor

- If a heavy load places on the motor, the driver will increase the current to the motor coil as it attempts to rotate the motor. Basically, there is no out-of-step condition.
- High-speed operation is possible.

4.5 Disadvantages of Servo Motor

- Since the motor tries to rotate according to the command pulses, but lags behind. it is not suitable for precision control of rotation.
- Higher cost.
- When stopped, the motor's rotor continues to move back and forth one pulse. So that, it is not suitable if you need to prevent vibration

4.6 Applications of Servo Motor

It uses in the applications requiring rapid variations in speed without the motor getting overheated.

- Industries, they uses in the machine tools, packaging, factory automation, material handling, printing converting, assembly lines. In many other demanding applications robotics, CNC machinery or automated manufacturing.
- uses in radio controlled airplanes to control the positioning and movement of elevators.
- In robots because of their smooth switching on and off and accurate positioning.
- In the aerospace industry to maintain hydraulic fluid in their hydraulic systems.
- uses in many radio controlled toys.
- used in electronic devices such as DVDs or Blue ray Disc players to extend or replay the disc trays.
- used in automobiles to maintain the speed of vehicles

5. **Driver Motor**
   - 5.1 **L293D**
   - 5.2 **L298N**
   - 5.3 **BTS7960N**
   - 5.4 **A4988**
   - 5.5 **TB6600**
   - 5.6 **PCA9685**
6. **ARM_ROBOT**
7. Spring-mass (suspension)

VI Sensor

1. **Encoder (check Velocity)**
   - 1.1 **What is Encoder?**
     An encoder is a sensing device that provides feedback. Encoders convert motion to an electrical signal that can be read by some type of control device in a motion control system, such as a counter or PLC. The encoder sends a feedback signal that can be used to determine position, count, speed, or direction. A control device can use this

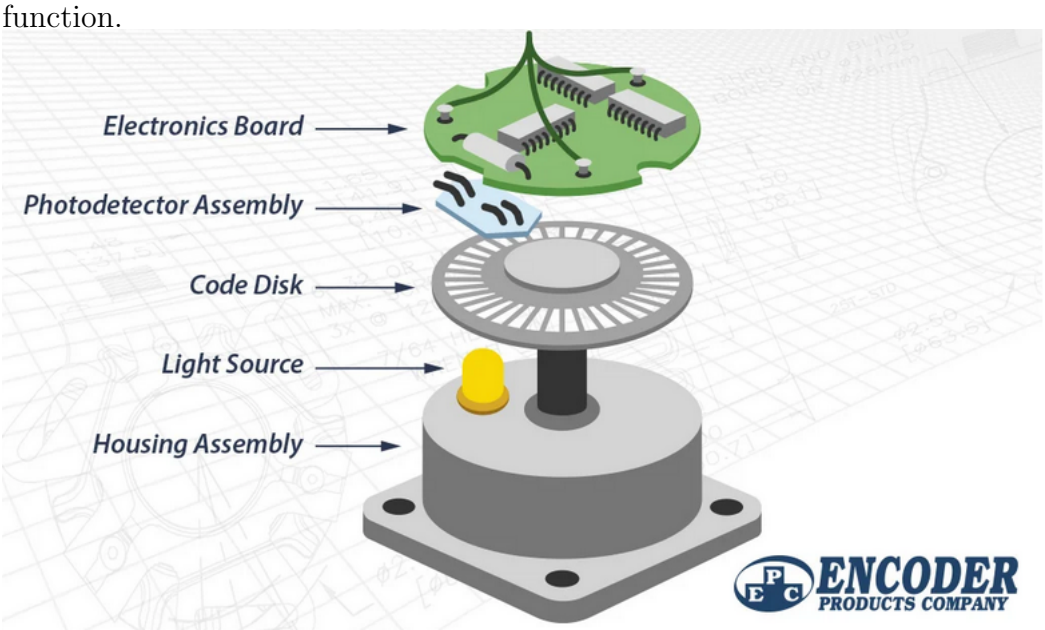
information to send a command for a particular function.
Example:

- In a cut-to-length application: an encoder with a measuring wheel tells the control device how much material has been fed,so the control device knows when to cut.
- In an observatory: the encoders tell actuators what position a moveable mirror is in by providing positioning feedback.
- On railroad-car lifting jacks: precission-motion feedback is provide by encoder,so the jacks lift in unison.
- In a precision servo label application system: the encoder signal is used by the PLC to control the timing and speed of bottle rotation.

1.2 How does an encoder work?

Encoders use different types of technologies to create a signal, including: mechanical, magnetic, resistive and optical being the most common. In optical sensing, the encoder provides feedback based on the interruption of light.

The graphic below outlines the basic construction of an incremental rotary encoder using optical technology. A beam of light emitted from an LED passes through the Code Disk, which is patterned with opaque lines (much like the spokes on a bike wheel). As the encoder shaft rotates, the light beam from the LED is interrupted by the opaque lines on the Code Disk before being picked up by the Photodetector Assembly. This produces a pulse signal: light = on; no light = off. The signal is sent to the counter or controller, which will then send the signal to produce the desired function.



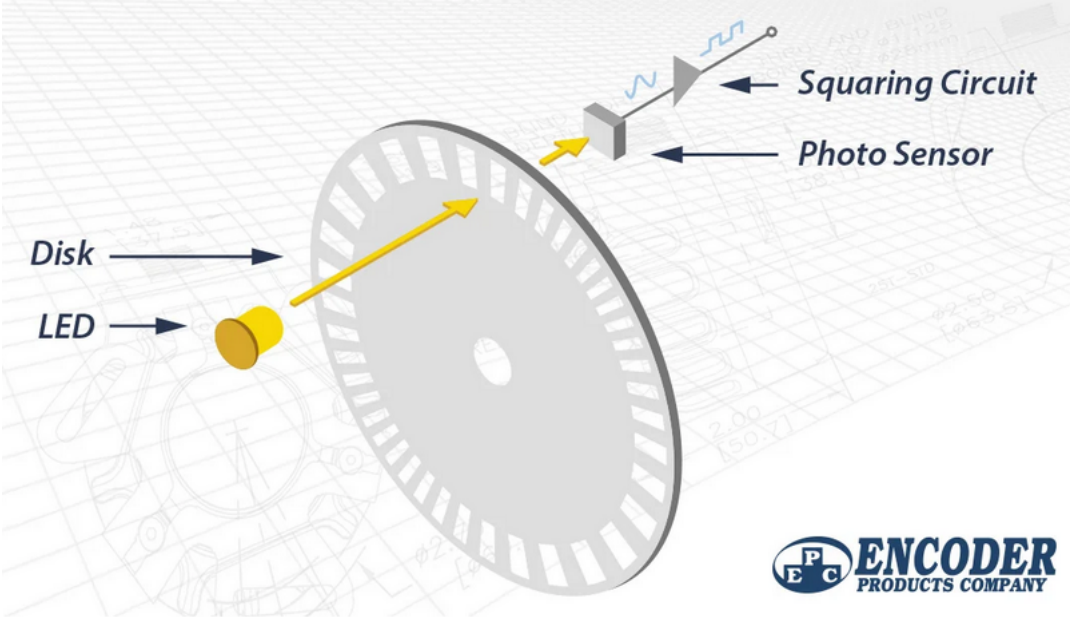
1.3 What's the difference between absolute and incremental encoders?

Encoders may produce either incremental or absolute signals.Incremental signals do not indicate specific position,only that the position has changed. Absolute encoders, on the other hand, use a different "word" for each position, meaning that an absolute encoder provides both the indication that the position has changed and an indication of the absolute position of the encoder.

1.4 What is an Optical Encoder?

An OPTICAL ENCODER IS A TYPE OF MOTION SENSING DEVICE that uses light shone through a coded disk to track the movement of a shaft.The encoder providees feedback based on the interruption of light.
Broken down in steps,this is how an optical encoder provide motion feedback infor-

mation:
1. A beam of light emitted from an LED passes through a code Disk, a transparent disk patterned with opaque lines (much like the spokes on a bike wheel).
2. The light beam is picked up by a Photodetector Assembly,also called a photodiode array or a photosensor.
3. The Photodetector Assembly responds to the light beam,producing a sinusoidal wave form,which is transformed into a square wave or pulse train.
4.This pulse signal is simple:
light=on
no light =off
5. The pulse signal is then sent to the counter or controller through the Electronics Board.
6. The counter or controller (not pictured)then sends the signal to produce the proper function (stop,go,rotate,etc.).



2. **IMU (check Angle and Accelerometer)**

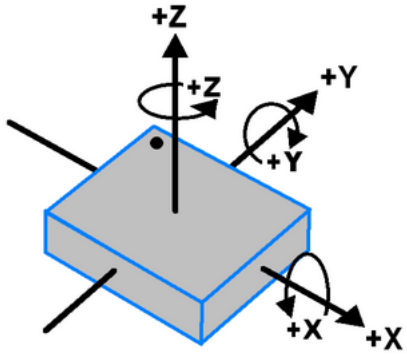
2.1 **MPU6050**

2.2 **What is MPU6050?**

MPU6050 sensor module is complete 6-axis Motion Tracking Device.It combines 3-axis Gyroscope,3-axis Accelerometer and Digital Motion Processor all in small package.
It has Auxiliary I2C bus to communicate with other sensor devices like 3-axis Magnetometer, Pressure sensor etc.

2.3 **3-Axis Gyroscope:(roll,pitch,yaw)**

It is used to detect rotational velocity along the X, Y, Z axes as shown in below figure.



MPU-6050
Orientation & Polarity of Rotation

- When the gyros are rotated about any of the sense axes,the Coriolis Effect causes a vibration that is detected by a MEM(Micro Electro Mechanical System) inside MPU6050.
- the resulting signal is amplified,demodulated,and filtered to produce a voltage that is propotional to the angular rate.
- This voltage is digitized using 16-bit ADC to sample each axis.
- The full-scale range of output are +/- 250, +/- 500, +/- 1000, +/- 2000 degrees.
- It measures the angular velocity along each axis in degree per second unit.
- Sensitivity of 131, 65.5, 32.8, or 16.4 LSBs per dps.
- Output data rate (ODR) range of 8kHz to 1.25Hz

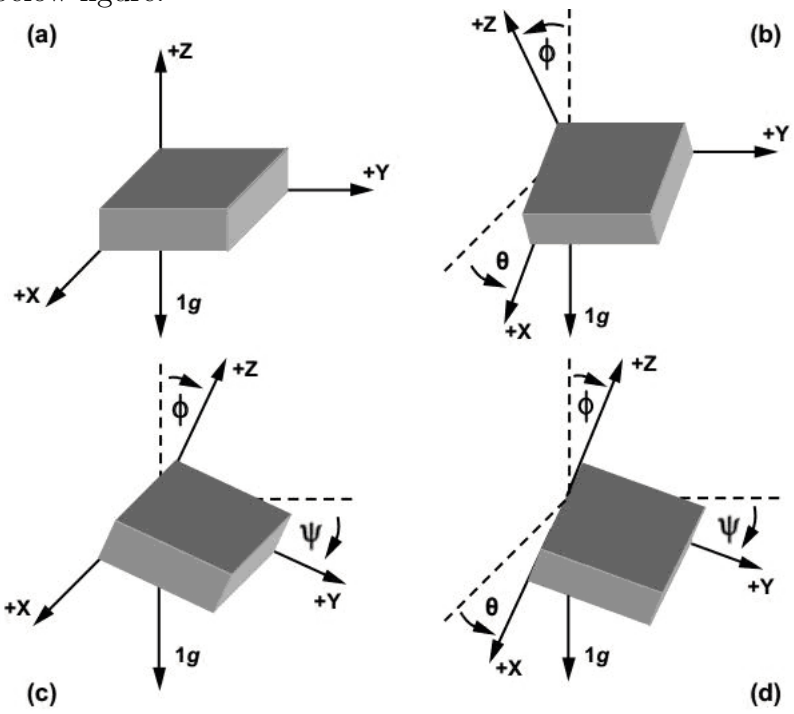
+Calculate:

Gyroscope full scale range of +/-250°/s with sensitivity scale factor of 131 LSB (count)/°/s:

⇒ Angular velocity along the (X,Y,Z) axis = (Gyroscope(X,Y,Z) axis raw data/131)(°/s)

2.4 3-Axis Accelerometer

It used to detect angle of tilt or inclination along the X, Y and Z axes as shown in below figure.



- Acceleration along the axes deflects the movable mass.
- This displacement of moving plate (mass) unbalances the differential capacitor which results in sensor output. Output amplitude is proportional to acceleration.

+Calculate:
Accelerometer full scale range of +/-2g with sensitivity scale factor of 16384 LSB (count)/g:
⇒ **Acceleration along the (X,Y,Z) axis = (Accelerometer(X,Y,Z) axis raw data/16384)g**

2.5 Combine agyroscope and accelerometer to measure angle

- $\mathbf{Angle_{pitch}} = \int_0^{k-T_s} \mathbf{Rate_{pitch}}.dt$
  ⇒ $\mathbf{Angle_{pitch}(k)} = \mathbf{Angle_{pitch}(k-1)} + \mathbf{Rate_{pitch}(k)}.\mathbf{Ts}$
- $\mathbf{Angle_{kalman}(k)} = \mathbf{Angle_{kalman}(k-1)} + \mathbf{Ts}.\mathbf{Rate(k)}$
  - rotation rate [°/s] measured by the gyroscope
  - Kalman: prediction of the angle,but not the final value.
- $\mathbf{Uncertainty_{angle}(k)} = \mathbf{Uncertainty_{angle}(k-1)} + \mathbf{Ts^2}.\mathbf{4^2}$
  Uncertainty fo the angle prediction (std dev of rate (k) = 4°/s
- $\mathbf{Angle_{kalman}(k)} = \mathbf{Angle_{kalman}(k)} + \mathbf{Gain_{kalman}}.(\mathbf{Angle(k)} - \mathbf{Angle_{kalman}})$
  - angle[°] measured by the accelerometer
  - $\mathbf{Gain_{kalman}} = \frac{\mathbf{Uncertainty_{angle}(k)}}{\mathbf{Uncertainty_{angle}(k)+3^2}}$ ((Std dev angle(k)=3°(acceterometer)
  - $\mathbf{Uncertainty_{angle}(k)} = (\mathbf{1} - \mathbf{Gain_{kalman}}) - \mathbf{Uncertainty_{angle}(k)}$
- **Predict the current state of the system:**
  $\mathbf{S(k)} = \mathbf{F}.\mathbf{S(k-1)} + \mathbf{G}.\mathbf{U(k)}$
  Where S=state vector ($Angle_{kalman}$; F=state transition matrix; G=Control matrix(0.004); U=input variable(Rate)
- **Calculate the Uncertainty of the prediction:**
  $\mathbf{P(k)} = \mathbf{F}.\mathbf{P(k-1)}.\mathbf{F^T} + \mathbf{Q}$
  Where:P=prediction uncertainty vector($Uncertainty_{angle}$)
  Q=Process uncertainty ($Ts^2.4^2$)
- **Calculate the Kalman gain from the uncertainties on the predictions and measurements:**
  $\mathbf{L(k)} = \mathbf{H}.\mathbf{P(k)}.\mathbf{H^T} + \mathbf{R}$

  $\mathbf{K} = \mathbf{P(k)}.\frac{\mathbf{H^T}}{\mathbf{L(k)}}$
  Where L=Intermediate matrix ; K=kalman gain;H=observation matrix(=1)
  R=Measurement uncertainty($Ts^2.3^2$)
- **Update the predicted State of the system with the measurement of the state through the kalman gain:**
  S(k)=S(k-1)+K.(M(K)-H.S(k))
  Where M=measurement vector (Angle)
- **update the uncertainty of the predicted state:**
  P(k)=(I-K.F).P(k-1)
  Where I=unitymatrix(=1)

2.6 Bno055

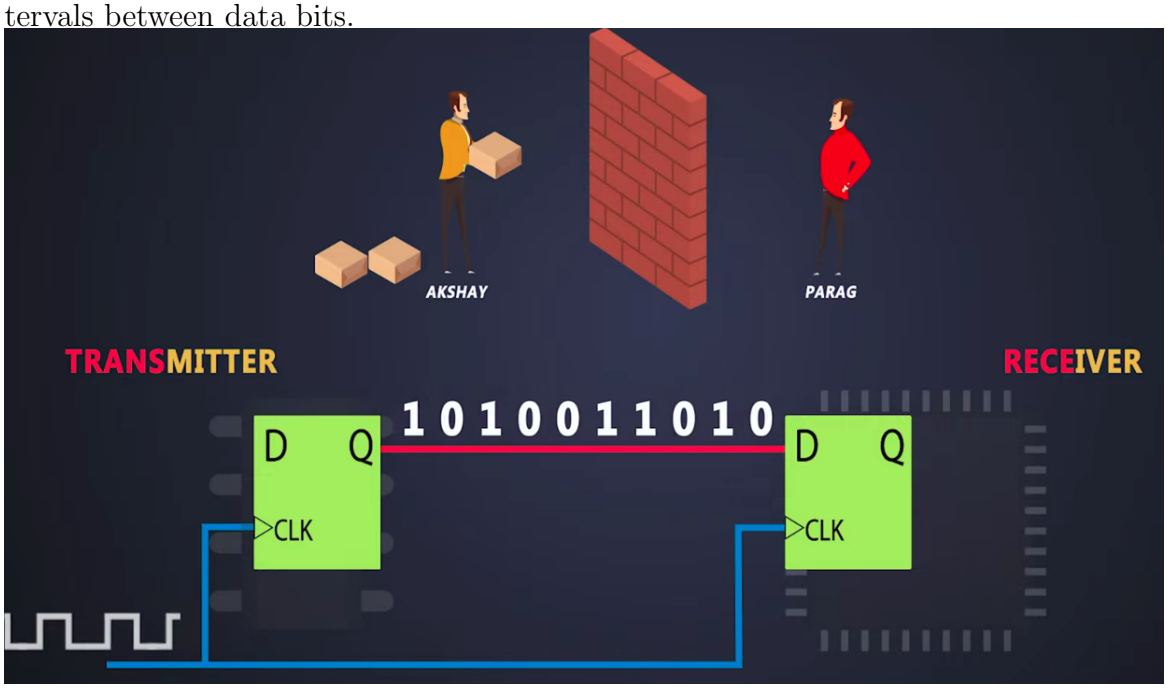
3. **Wheel Rotary Encoder**
4. **Limit_Switch (check limit)**
5. **Camera**
6. **Laser (check Distance)**

VII Communication

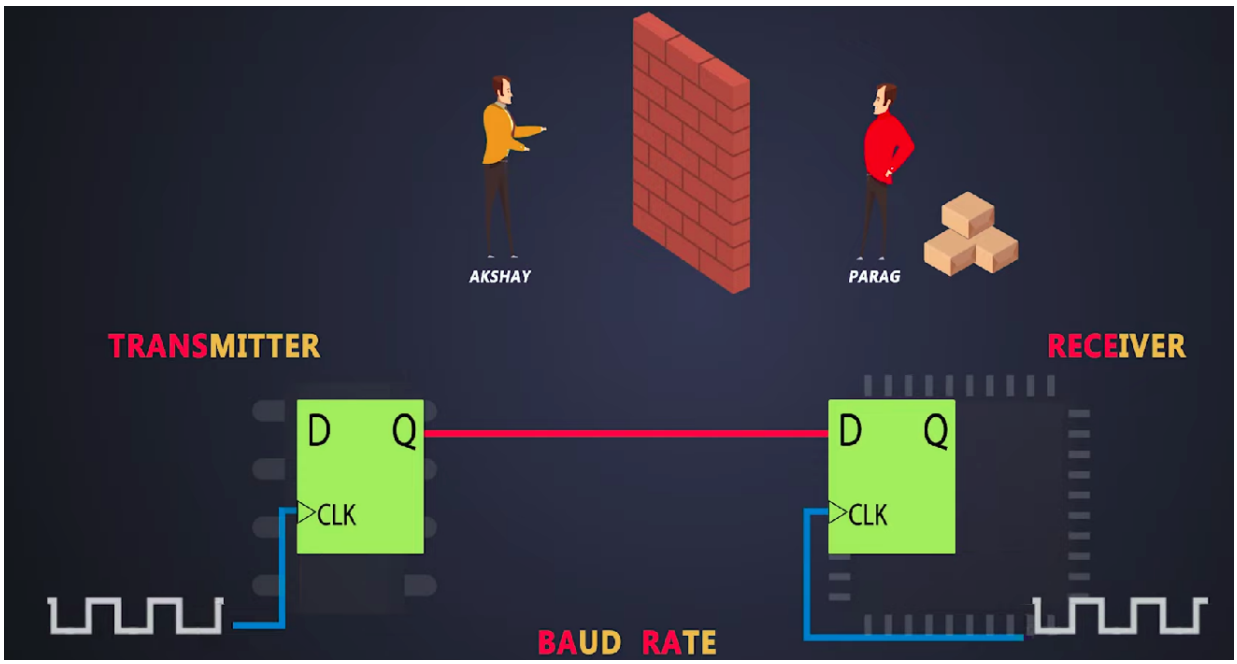
What is communicate protocol?
A communication protocol is a system of rules that allows two or more entities of a communications system to transmit information via any variation of a physical quantity.
Communication protocols are formal descriptions of digital message formats and rules. They are required to exchange messages in or between computing systems. Communication protocols are important in telecommunications systems and other systems because they create consistency and universality for the sending and receiving of messages.

1. **Synchronous & Asynchronous communication**

   1.1 **Synchronous communication protocol**
   Synchronous is a communications protocol used to send and receive data across high-speed main frame networks.Data is synchronized and transmitted using constant intervals between data bits.



   1.2 **Asynchronous communication protocol**
   Asynchronous is a communication protocol that transmits one byte of in formation at a time at irregular time intervals.A start bit(one character)preceds the byte(eight characters) of information and a stop bit (one character) follow the byte.asynchronous communication is transmission of data, generally without the use of an external clock signal.



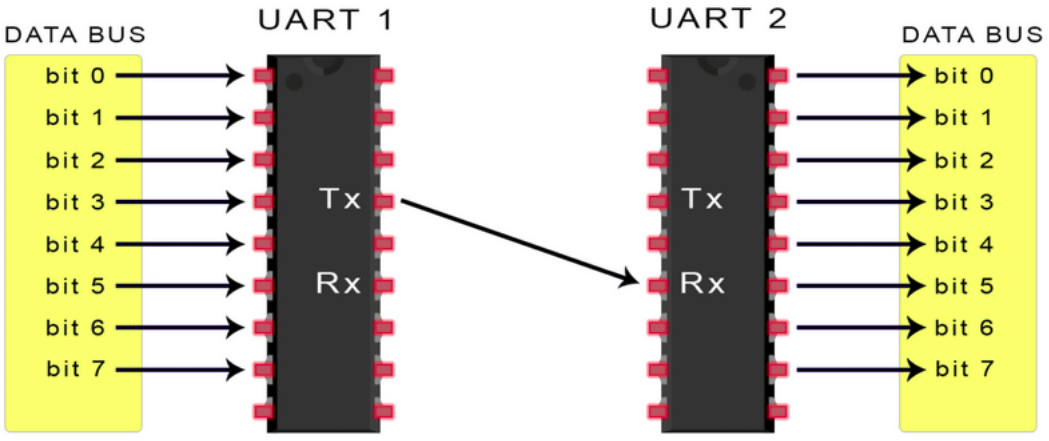
2. UART(Universal Asynchronous Receiver/Tramsmitter)

2.1 What is UART?

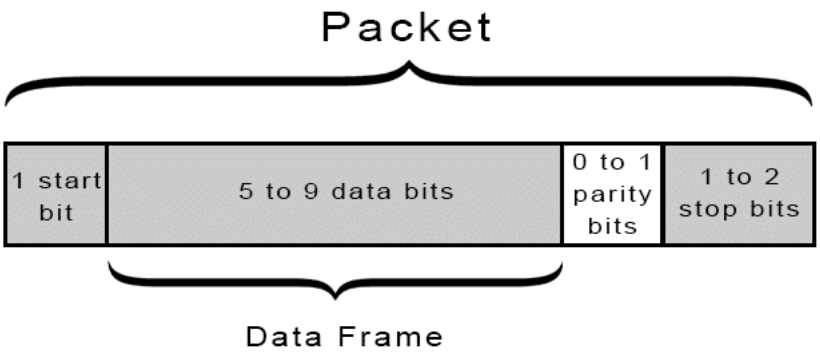
UART is hadware communication protocol that uses asynchronous serial communication with configurable speed.

2.2 How UART works

- The UART that is going to transmit data receives the data from a data bus
- The data bus is used to send data to the UART by another device like a CPU,memory,or microcontroller.
- Data is transferred from the data bus to the transmitting UART in parallel form.
- After the transmitting UART gets the parallel data from the data bus, it adds a start bit, a parity bit, and a stop.
- Next, the data packet is output serially, bit by bit at the Tx pin,The receiving UART reads the data packet bit by bit at its Rx pin.
- The receiving UART then converts the data back into parallel form and removes the start bit, parity bit, and stop bits.
- Finally, the receiving UART transfers the data packet in parallel to the data bus on the receiving end.



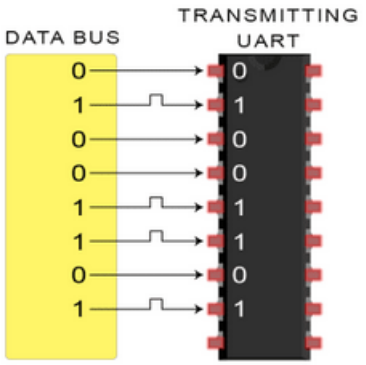
Packet



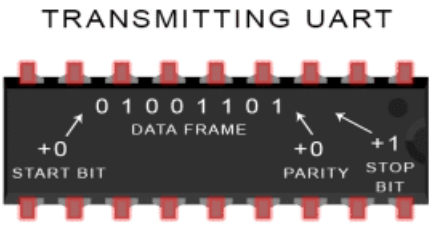
- **Start Bit:**
  The UART data transmission line is normally held at a high voltage level when it's not transmission line from high to low for one clock cycle.When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate.
- **Data Frame:**
  The data frame contains the actual data being transferred. It can be 5 bits up to 8 bits long if a parity bit is used. If no parity bit is used, the data frame can be 9 bits long. In most cases, the data is sent with the least significant bit first.
- **Parity:**
  Parity describes the evness or oddness of a number.The parity bit is away for the receiving UART to tell if any data has changed during transmission.Bits can be changed by electromagnetic radiation, mismatched baud rates,or long distance data transfers.After the receiving UART reads the data frame,it counts the number of bits with avalue of 1 and checks if the total is an even or odd number.If the parity bit is a 0 (even parity),the 1 bits in the data frame should total to an even number. If the parity bit is a 1 (odd parity), the 1 bits in the data frame should total to an odd number. When the parity bit matches the data, the UART knows that the transmission was free of errors. But if the parity bit is a 0, and the total is odd; or the parity bit is a 1, and the total is even, the UART knows that bits in the data frame have changed.
- **Stop Bits:**
  To signal the end of the data packet,the sending UART drives the data transmission line from a low voltage to a high voltage for at least two bit durations.

2.3 Steps of UART Transmission

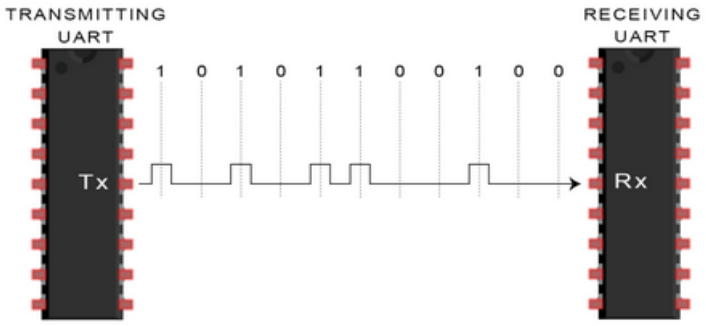
- **Step1:** The transmitting UART receives data in parallel from the data bus:



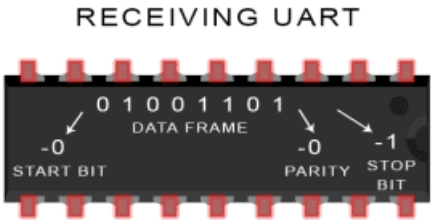
- **Step2:** The transmitting UART adds the start bit,parity bit,and the stop bit(s) to the data frame:



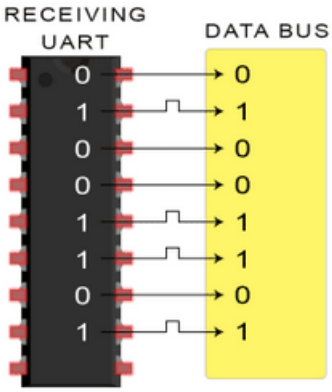
- **Step3:** The entire packet is sent serially from the transmitting UART to the receiving UART.The receiving UART samples the data line at the preconfiguted baud rate:



- **Step4:** the receiving UART discards the start bit, parity bit, and stop bit from the data frame:



- **Step5:** The receiving UART converts the serial data back into parallel and transfers it to the data bus on the receiving end:



2.4 Advantages and Disadvantages of UARTs:

Advantages

- Only uses two wires
- No clock signal is nescessary
- Has a parity bit to allow for error checking
- The structure of the data packet can be changed as long as both sides are set up for it
- Well documented and widely used method

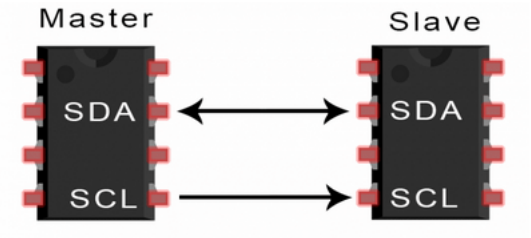
Disadvantages

- The size of the data frame is limited to a maximum of 9 bits.
- Doesn't support multiple slave ot multiple master systems.
- The baud rates of each UART must be within 10% of each other.

3. **I2C(Inter-integrated circuits)**

3.1 **What is I2C?**

_I2C is a serial communication protocol,so data is transferred bits by bit along a single wire (the SDA line).like SPI,I2C is synchronous , so the output of bit is synchronized to the sampling of bits by a clock signal shared between the master and the slave .The clock signal is always controlled by the master.
_You'll probably find yourself using I2C if you ever build projects that use OLED displays, barometric pressure sensors, or gyroscope/accelerometer modules.
_I2C only uses two wires to transmit data between devices:

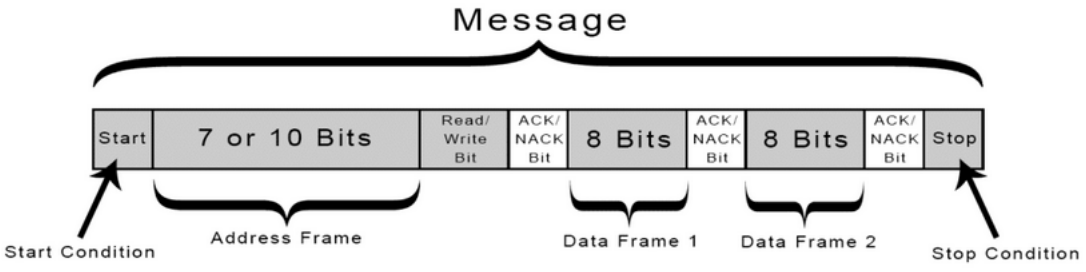


Serial Data(SDA): The line for the master and slave to send and receive data.
Serial Clock(SCL): The line that carries the clock signal.

Wires Used	2
Maximum Speed	Standard mode= 100 kbps
	Fast mode= 400 kbps
	High speed mode= 3.4 Mbps
	Ultra fast mode= 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	Unlimited
Max # of Slaves	1008

3.2 **How I2C Works**

With I2C, data is transferred in messages. Messages are broken up into frames of data:



Start Condition:

- To initiate the address frame,the master device leaves SCL high and pulls SDA low.
- This puts all slave devices on notice that a transmission is about to start.
- If two master devices wish to take owner shop of the bus at one time,which ever device pull SDA low first wins the race and gains control of the bus.

Address Frame:

- For a 7-bit address,the address is clockedout most significant bit (MSB) first,followed by a R/W bit indicating whether this is a read(1) or write(0) operation.
- A 7 or 10 bit sequence unique to each slave that identifies the slave when the master wants to talk to it.

Read/Write Bit:

A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).

ACK/NACK Bit:

Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

Data Frame:

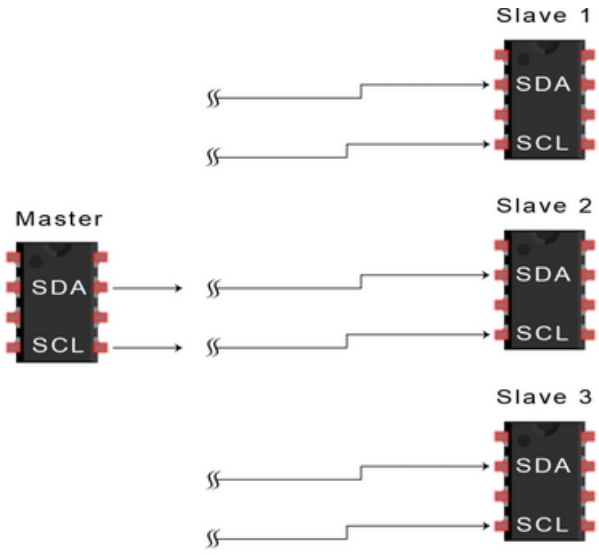
The master will simply continue generating clocks pulses at a regular interval,and the data will be placed on SDA by either the master or the indicated a read or write operation.

Stop condition:

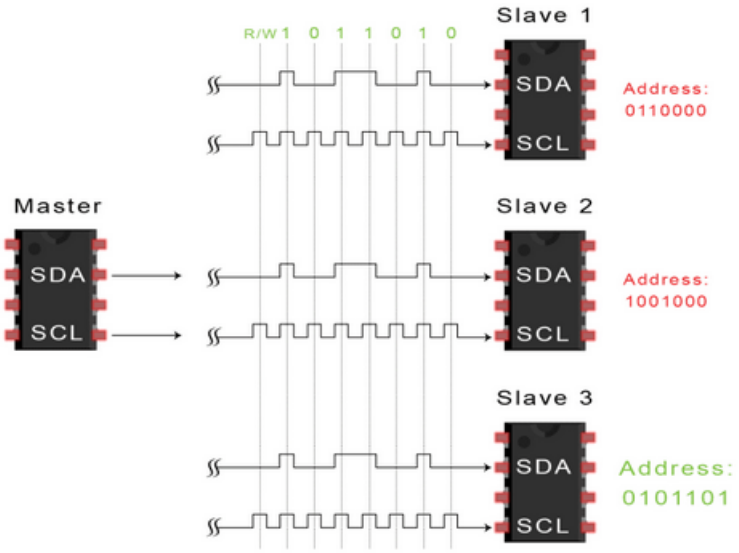
Stop conditions are defined by a 0→1 (low to high)transition on SDA after a 0→1 transition on SCL,with SCL remaining high.

3.3 Steps of I2C Data Transmission

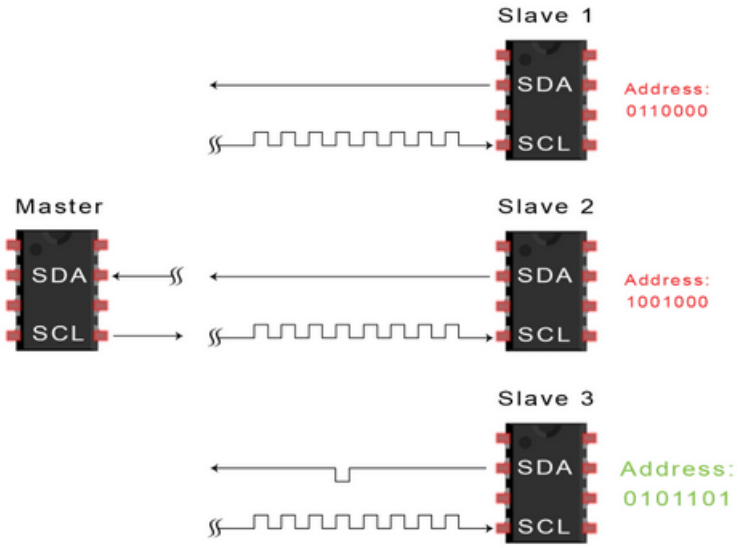
- **Step1:** The master sends the start condition to every connected slave by switching the SDA line from a high voltage level to a low voltage level before switching the SCL line from high to low:



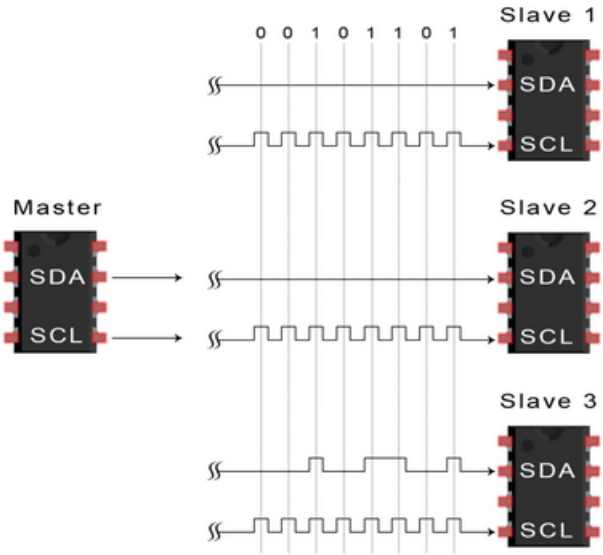
- **Step2:** The master sends each slave the 7 or 10 bit address of the slave it wants to communicate with along with the read/write bit:



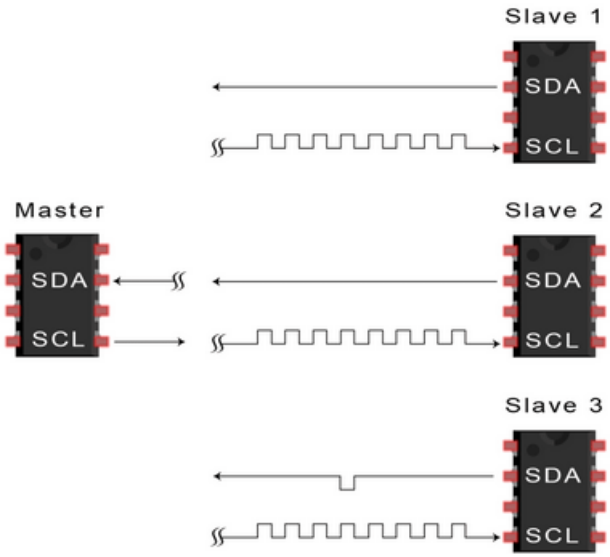
- **Step3:** Each slave compares the address sent from the manster to its own address.If the address matches,the slave returns an ACK bit by pulling the SDA line low for one bit.If the address from the master does not match the slave's own address,the slave leaves the SDA line high.



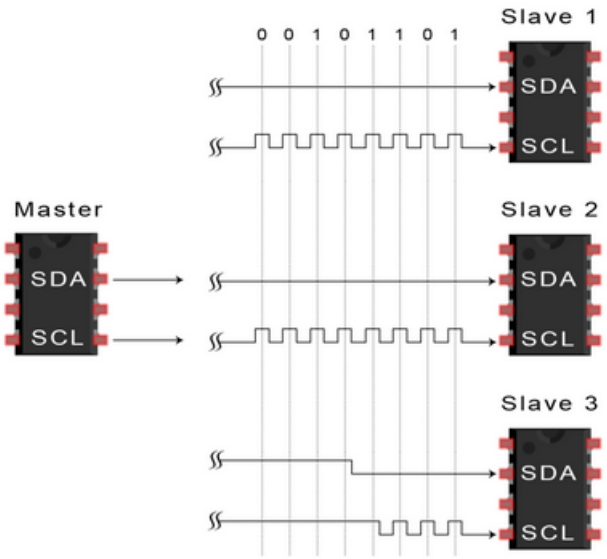
- **Step4:** The master sends or receives the data frame:



- **Step5:** After each data frame has been transferred,the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame:

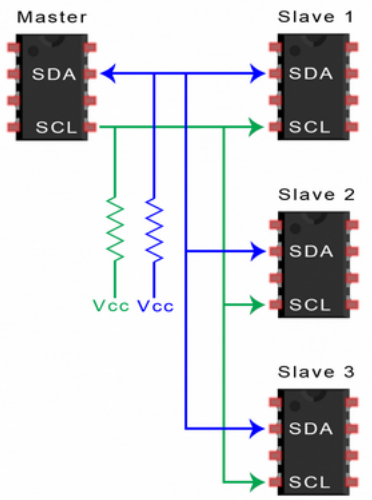


- **Step6:** To stop the data transmission,the master sends a stop condition to the slave by switching SCL high before switching SDA high:



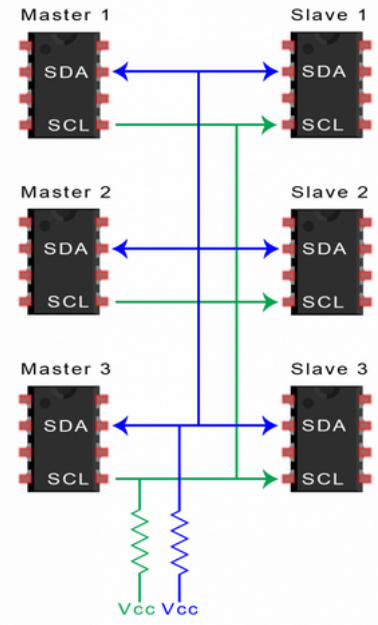
3.4 Single Master with Multiple Slaves

Because I2C uses addressing,multiple slaves can be controlled from a single master. With a 7 bit address, 128 (27) unique address are available. Using 10 bit addresses is uncommon, but provides 1,024 (210) unique addresses. To connect multiple slaves to a single master, wire them like this, with 4.7K Ohm pull-up resistors connecting the SDA and SCL lines to Vcc:



3.5 Multiple Masters with Multiple Slaves

Multiple master can be connected to a single slave ot multiple slaves.The problem with multiple master in the same system comes when two masters try to send or receive data at the same time over the SDA line. To solve this problem, each master needs to detect if the SDA line is low or high before transmitting a message. If the SDA line is low, this means that another master has control of the bus, and the master should wait to send the message. If the SDA line is high, then it's safe to transmit the message. To connect multiple masters to multiple slaves, use the following diagram, with 4.7K Ohm pull-up resistors connecting the SDA and SCL lines to Vcc:



3.6 Advantages and Disadvantages of I2C

There is a lot of to I2C that might make it sound complicated compared to other protocols, but there are some good reasons why you may or may not want to use I2C to connect to a particular device:

ADVANTAGES

- Only uses two wire
- Supports multiple masters and multiple Slaves
- ACK/NACK bit gives confirmation that each frame is transferred successfully
- Hardware is less complicated than with UARTs
- Well know and widely used protocol

DISADVANTAGES

- Slower data transfer rate than SPI
- The size of the data frame is limited to 8 bits
- More complicated hardware needed to implement than SPI

4. **SPI(Serial Peripheral Interface)**
SPI, I2C, and UART are quite a bit slower than protocols like USB, ethernet, Bluetooth, and WiFi, but they're a lot more simple and use less hardware and system resources. SPI, I2C, and UART are ideal for communication between microcontrollers and between microcontrollers and sensors where large amounts of high speed data don't need to be transferred.

4.1 What is SPI?

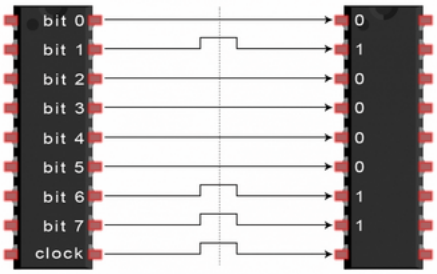
SPI is a common communication protocol used by many different devices.
Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards.

4.2 Serial vs. Parallel Communication

Electronic devices talk to each other by sending bits of data through wires physically connected between devices. A bit is like a letter in a word, except instead of the 26 letters (in the English alphabet), a bit is binary and can only be a 1 or 0. Bits are transferred from one device to another by quick changes in voltage. In a system operating at 5 V, a 0 bit is communicated as a short pulse of 0 V, and a 1 bit is communicated by a short pulse of 5 V.

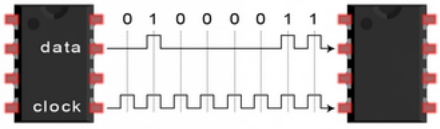
Parallel

In parallel communication, the bits of data are sent all at the same time, each through a separate wire. The following diagram shows the parallel transmission of the letter "C" in binary (01000011):

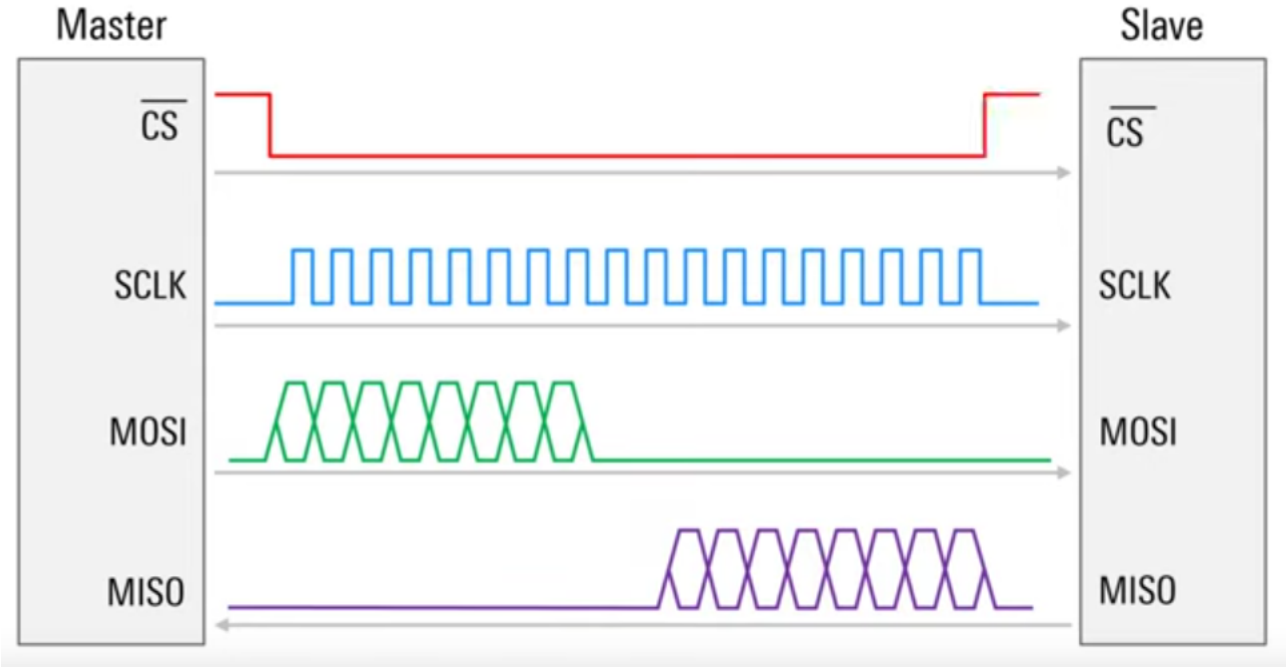


Serial

In serial communication, the bits are sent one by one through a single wire. The following diagram shows the serial transmission of the letter "C" in binary (01000011):



4.3 PINOUT



or SS:

Pulled low to select the target (slave) for commuincation:

- slave then listens for SCLK and MOSI
- Kept low until communication is complete

SCLK:

- Clock signal is generated by the master (Slave don't require their own clocks,even when they are transmitting data.
- Speed usually into the MHZ range (faster than UART or I2C)
- clock indicates when data should be sampled
  - when to read voltage levels
  - one bit is read per clock signal
- clock can be idle low or idle high
- Data can be sampled on rising or failing edge.

MOSI (Master Output/Slave Input):-

- Used to send data from master to slave,Data is usually sent as bytes (LSB or MSB first)
- Number of bytes depends on implementation.

MISO (Master Input/Slave Output):

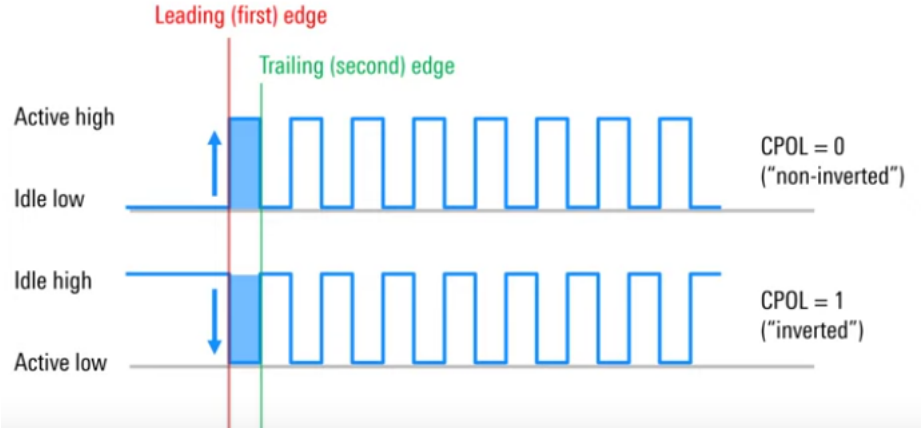
- Used to send data from slave (s) to master.
- Not all SPI implementation use MISO (Some devices only receive data from master)
- Master must know how many clock cycles to generate so the slave can send all of its data (May be known in advance or slave may be queried for this information).

Wires Used	4
Maximum Speed	Up to 10 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	1
Max # of Slaves	Theoretically unlimited*

4.4 HOW SPI WORKS The clock

- The clock signal synchronizes the output of data bits from the master to the sampling of bits by the slave.
- One bit of data is transferred in each clock cycle,so the speed of data transfer is determined by the frequency of the clock signal.
- SPI communication is always initiated by the master since the master configures and generates the clock signal.
- The clock signal in SPI can be modified using the properties of clock polarity and clock phase.

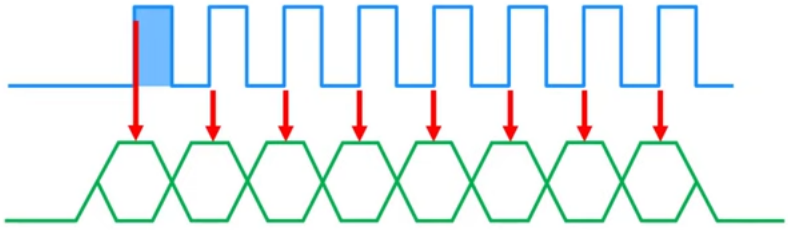
+CPOL(clock polarity):



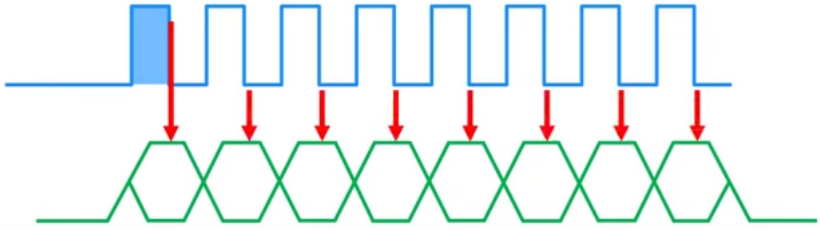
+CPHA(clock phase):

SPI receivers may sample data on either leading or trailing clock edge.

_CPHA=0: Data sampled on leading (or first) edge of each clock pulse



_CPHA=1: Data sampled on trailing (or second) edge of each clock pulse.

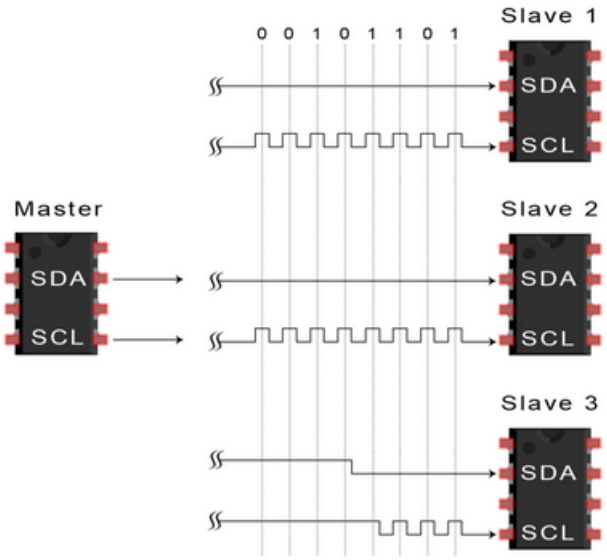


Slave Select

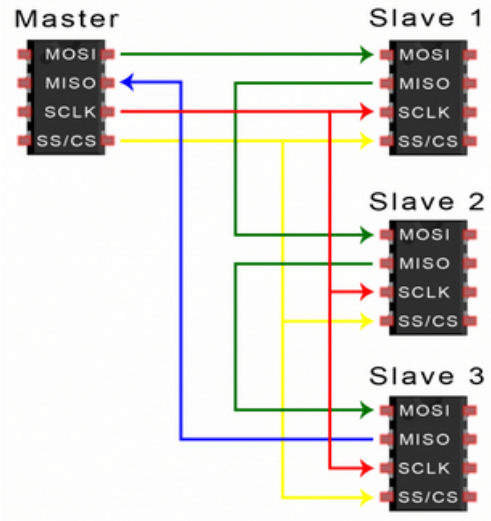
- The master can choose which slave it wants to talk to by setting the slave's CS/SS line to a low voltage level.
- In the idle, non-transmitting state, the slave select line is kept at a high voltage level.
- Multiple CS/SS pins may be available on the master, which allows for multiple slaves to be wired in parallel.
- If only one CS/SS pin is present, multiple slaves can be wired to the master by daisy-chaining.

Multiple Slaves

- SPI can be set up to operate with a single master and a single slave, and it can be set up with multiple slaves controlled by a single master.
- There are two ways to connect multiple slaves to the master.
- If the master has multiple slave select pins, the slaves can be wired in parallel like this:



If only one slave pin is available, the slaves can be daisy-chained like this:



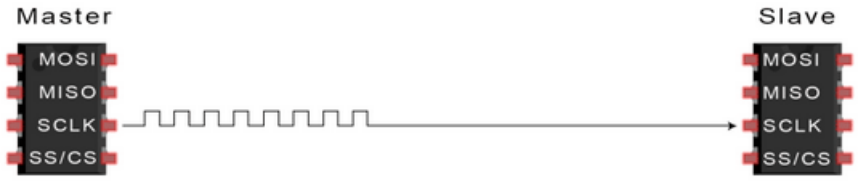
MOSI and MISO

- The master sends data to the slave bit by bit, in serial through the MOSI line.
- The slave receives the data sent from the master at the MOSI pin.
- Data sent from the master to the slave is usually sent with the most significant bit first.

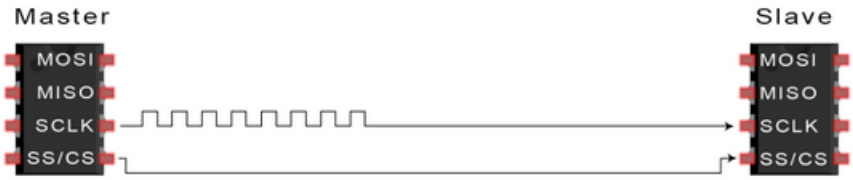
- The slave can also send data back to the master through the MISO line in serial.
- The data sent from the slave back to the master is usually sent with the least significant bit first.

4.5 Steps of SPI Data Transmission

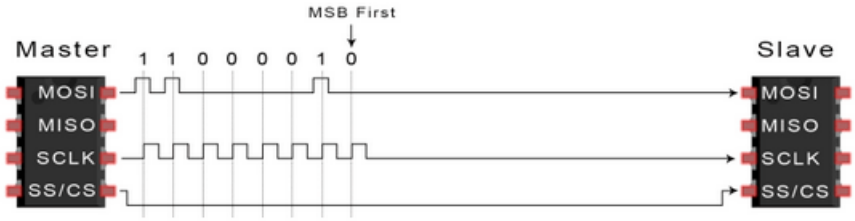
- **Step1:**The master outputs the clock signal:



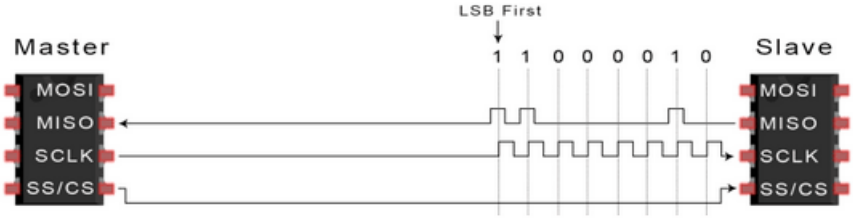
- **Step2:** The master switches the SS/CS pin to a low voltage state,which activates the slave:



- **Step3:** The master sends the data one bit at a time to the slave along the MOSI line.The slave reads the bits as they are received.



- Step4: If a response is needed,the slave returns data one bit at a time to the master along the MISO line .The master reads the bits as they are received:



4.6 Advantages and Disadvantages of SPI

Advantages:

- No start and stop bits,so the data can be streamed continuously without interruption
- No complicated slave addressing system like I2C
- Higher data transfer rate than I2C (almost twice as fast)
- Seperate MISO and MOSI lines,so data can be sent and received at the same time.

Disadvantages:

- Uses four wires(I2C and UARTs use two)
- No acknowledgement that the data has been successfully received (I2C has this)
- No form of error checking like the parity bit in UART
- Only allows for a single master

5. CAN(Control Area Network)

5.1 What is CAN(Control Area Network)?

The **Controller Area Network (CAN bus)** is a message-based protocol designed to allow the Electronic Control Units (ECUs) found in today's automobiles, as well as other devices, to communicate with each other in a reliable, priority-driven fashion. Messages or "frames" are received by all devices in the network, which does not require a host computer. CAN is supported by a rich set of international standards under ISO 11898.

5.2 Introduction

- Developed by Robert Bosch in 1986.
- No Host required.
- CAN bus is a broadcast type of bus.
- Message based protocol.
- Serial half-duplex Asynchronous Communicatio.
- Different Two wired communication.
- Designed originally for automobiles,but also used in many othe contexts.
- One of five protocols used in the on-board diagnostics.
- It is a multi-master Serial communication bus whose basic design specfication called for high speed,high noise-immunity and error-detection features
- CAN offers data communication up to 1Mbit/sec.

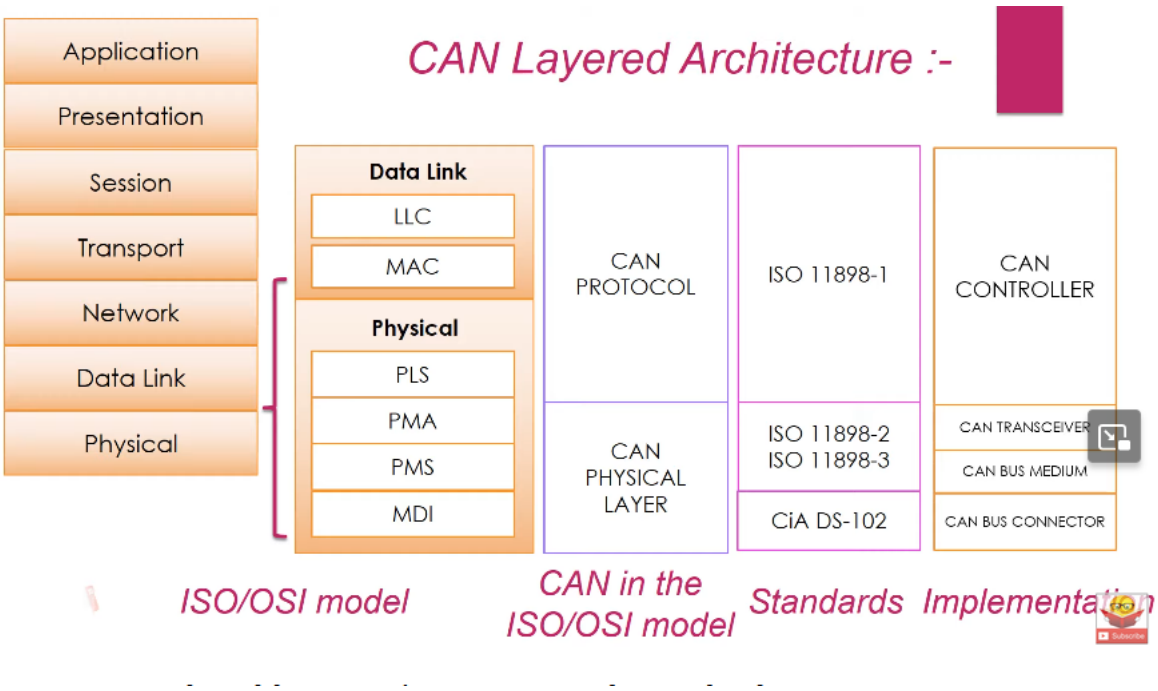
5.3 CAN's modt attrative features

- Low Cost
- Efficient
- Flexible
- Centralized
- Extreme robustness
- High data transmission speeds
- Reliability,Excellent error handling and Error confinement abilities.
- Automatic re-transmission of faulty messages.
- Automatic bus disconnection of nodes that are suspected to be physically faulty
- Functional addressing -data message do not contain source or destination address,only identifiers relating to their function and/or priority.

5.4 Applications CAN Protocol

- Automotive (Passenger vehicles,trucks,buses)
- Electronic equipment for aviation and navigation.
- Industrial automation and mechanical Control.
- Elevators & escalators.
- Building automation.
- Medial instruments and equipment.
- Marine,Military,Industrial,Medical

5.5 CAN Layered Architecture:



MAC: Medium Access Control.

- Medium Access
- Encapsulation/Decapsulation
- Error Detection
- Signalling

LLC: Logical Link Control.It is responsible for frame acceptance filtering ,overload notification,and recovery management.

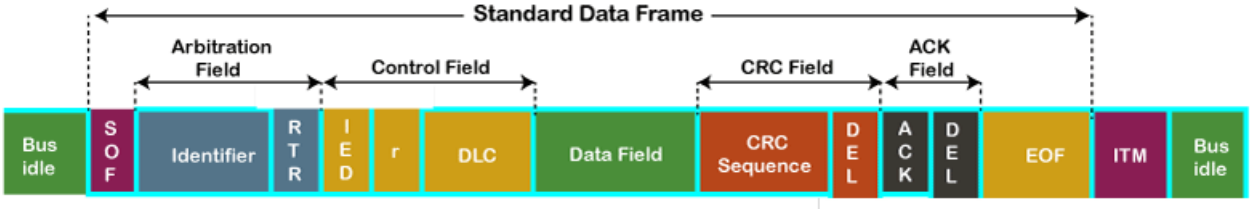
- Frame Acceptance Filtering
- Overload Notification
- Recovery Management

Physical layer: The physical layer is responsible for the transmission of raw data. It defines the specifications for the parameters such as voltage level, timing, data rates, and connector.

PLS: Physical Layer Signalling
PMA: Physical Medium Attachment
PMS: Physical Medium Specification.
MDI: Medium Dependent Interface

5.6 CAN Frame

Standard Frame:

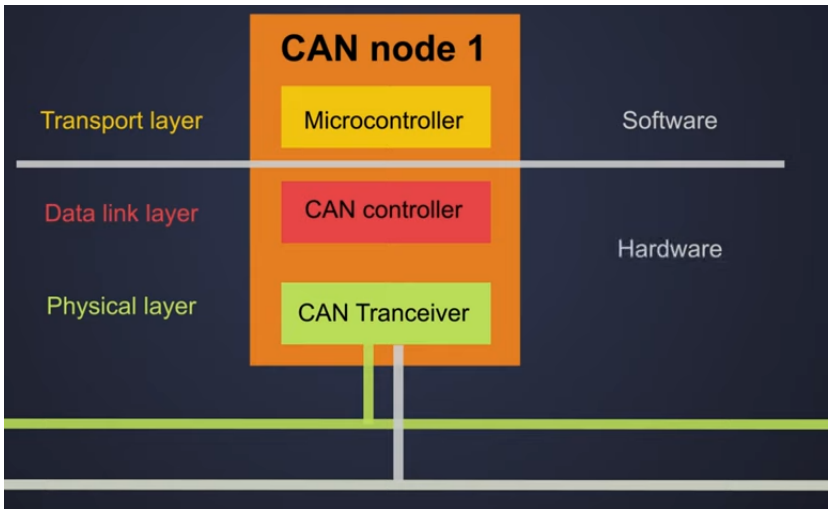


- **SOF:** The signal domaint start of frame (SOF) bit marks the start of a message and is used to synchronous the node on a bus after being idle .It's of 1 bit.
- **Identifier:** A standard data format defined under the CAN 2.0 A specification uses an 11-bit message identifier for arbitration. Basically, this message identifier sets the priority of the data frame.The lower the binary value,the higher is priority.

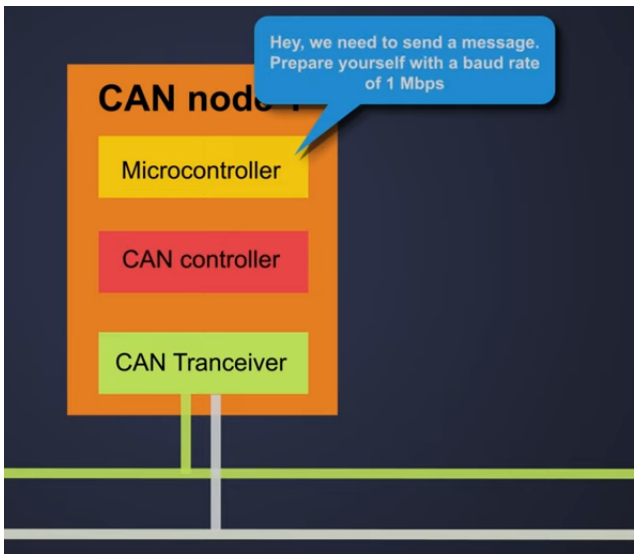
- **RTR:**
  _The single remote transmission request (RTR) bit is dominant when information is required from another node.
  _All node receive the request,but the identifier determines the specified node.
  _The responding data is also received by all nodes and by any node interested.
  _In this way,all DAta being used in a system is uniform.
- **Controller Field:**
  - **IDE:** A-dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.A dominant IDE bit defines the 11-bit standard identifier
  - **r:** Reserved bit (for possible use by future standard amendment).
  - **DLC:** The 4bit data length code (DLC) contains the number of bytes of data being transmitted.
- **Data Field:**Up to 8-bytes of application data may be transmited.
- **CRC Field:** The 16bit(15bits plus delimiter)cyclic redundancy check(CRC)contains the checksum (number of bits transmitted)of the preceding application data for error detection.
- **ACK Field:** Every node receiving an accurate message overwhites this recessive bit in the orginal message witht a dominate nit ,indicating an error-free message has been sent ,ACK is 2 bits,one is the acknowledgment bit and the second is a delimiter.If the CRC does not match, then the receiver will generate the error.
- EOF: This end of -frame (EOF),7bit field marks the end of a CAN frame (message) and esables bit stuffing ,indication as stuffing error when dominant when 5bit of the frame logic level occur in succession during normal operation,a bit of the opposite logic level is stuffed into data.

5.7 CAN Data Flow:

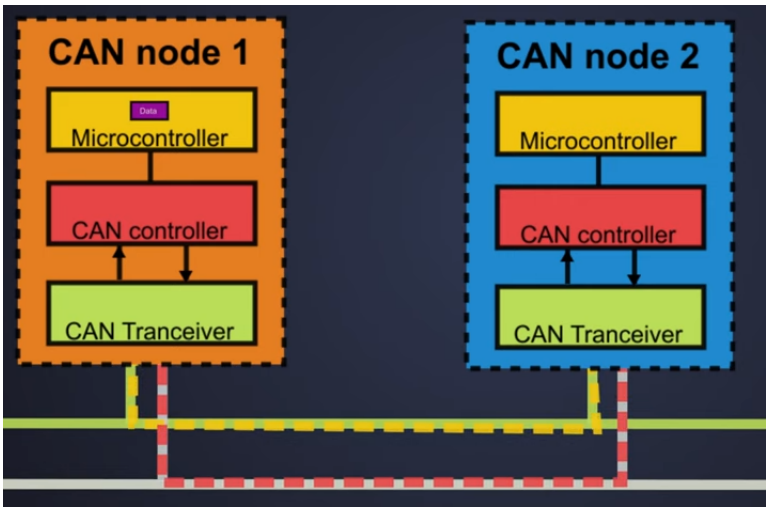
- What is CAN node?



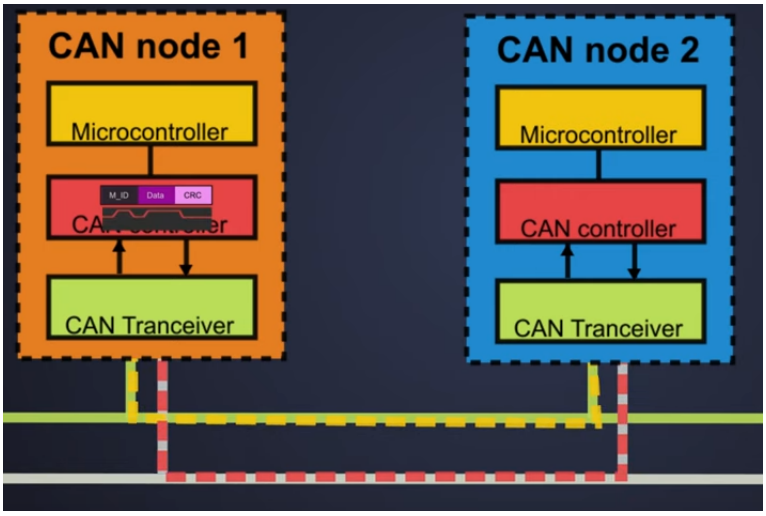
- CAN Data Transmission
  Step1:Set baud rate



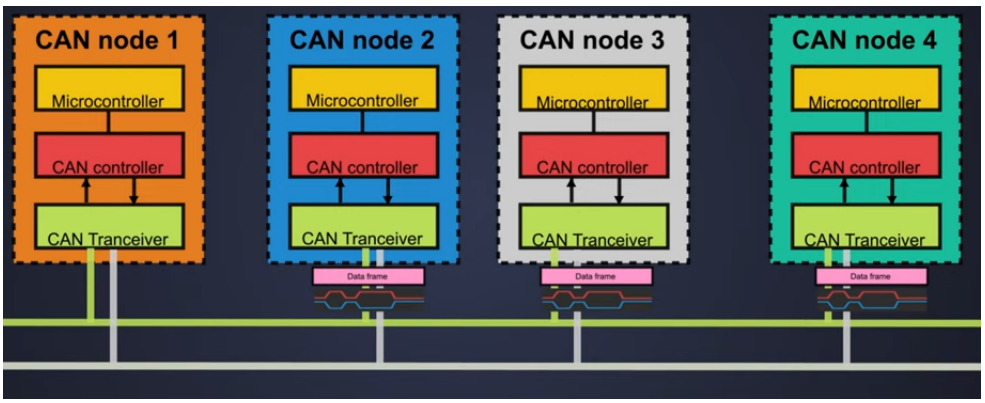
Step2:Set up Data



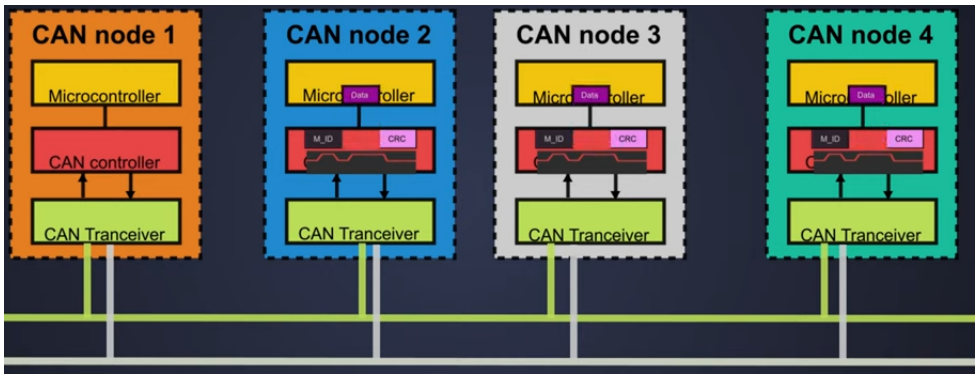
Step3:Set up to CAN_Control



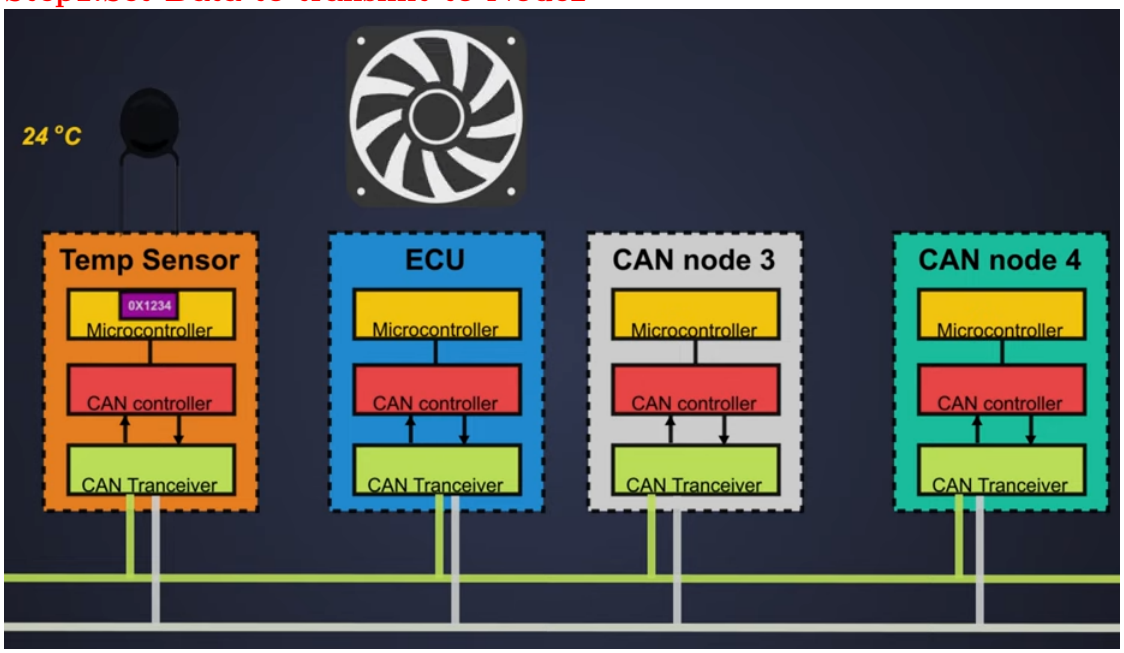
Step4:Transmit to each Node



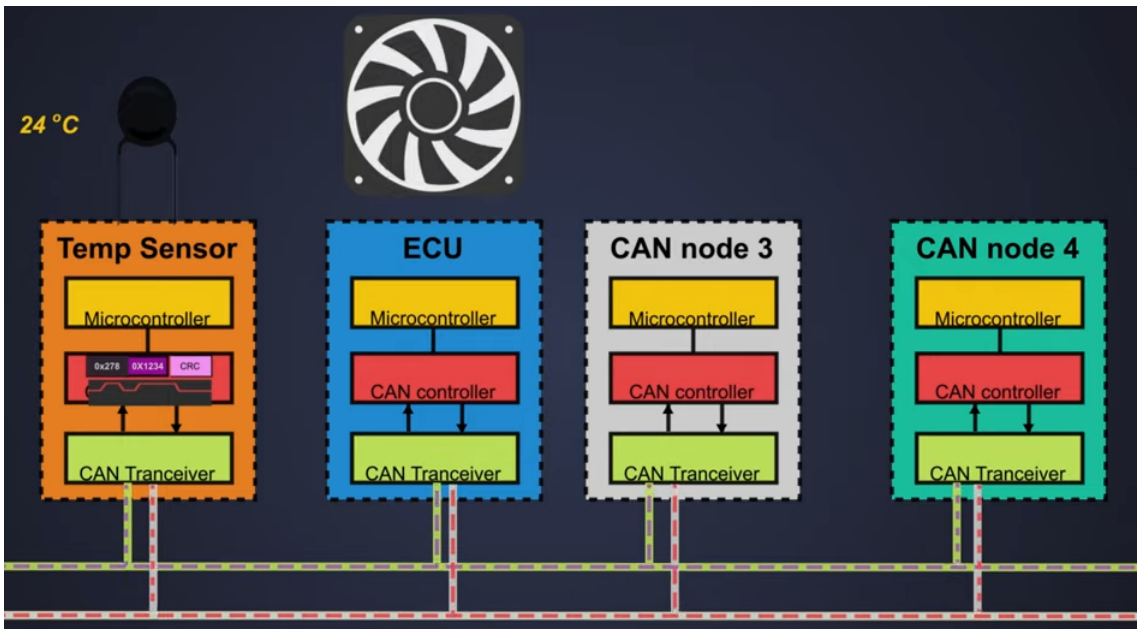
Step5:Data reception



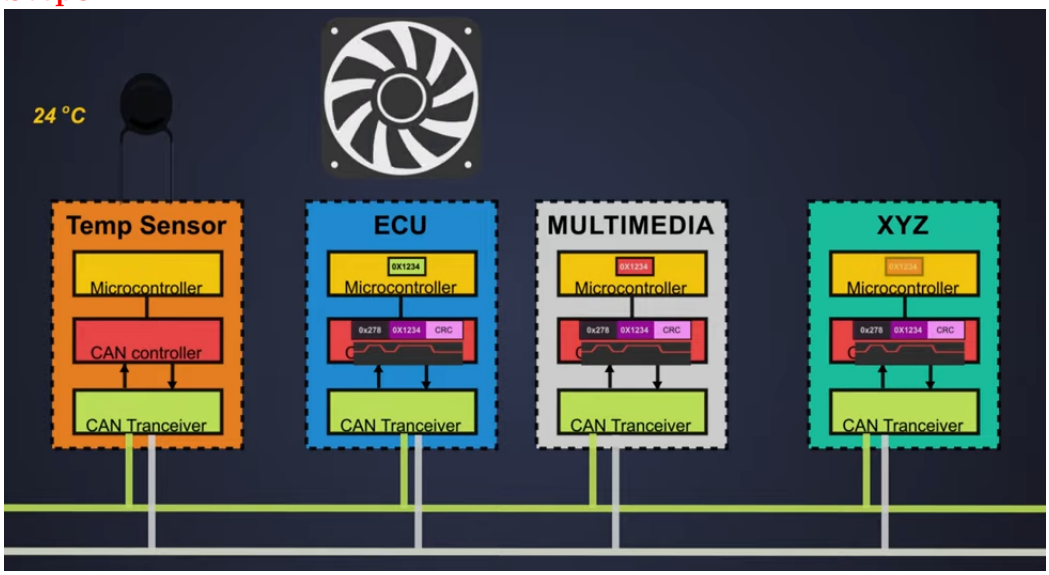
- **Example:**
  - **Step1:Set Data to transmit to Node2**



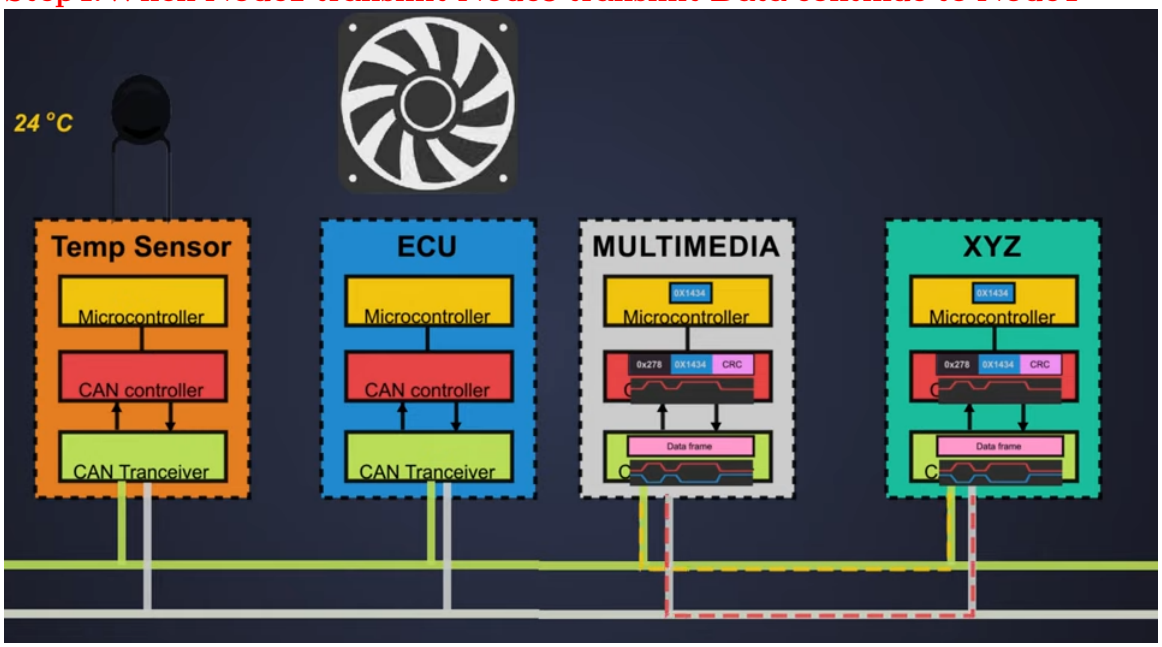
  - **Step2:**



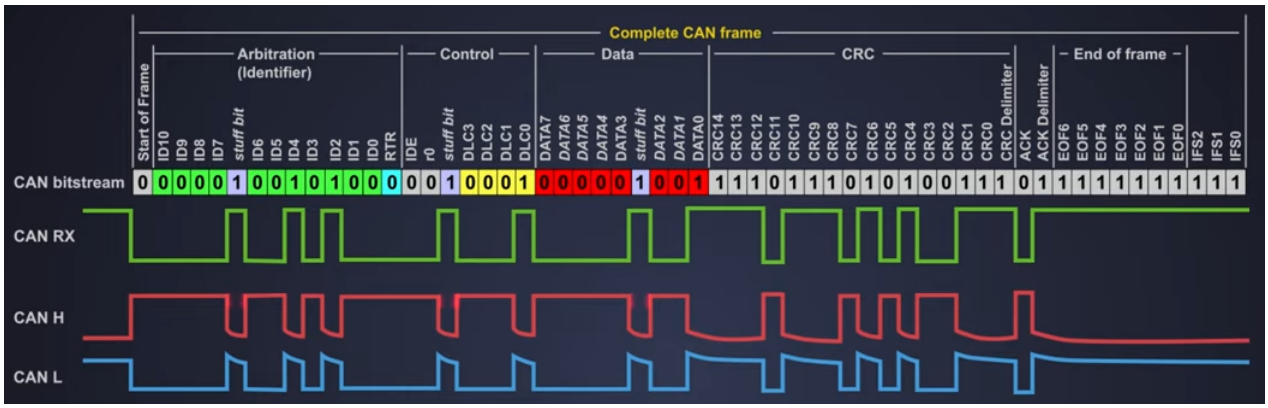
- **Step3:**



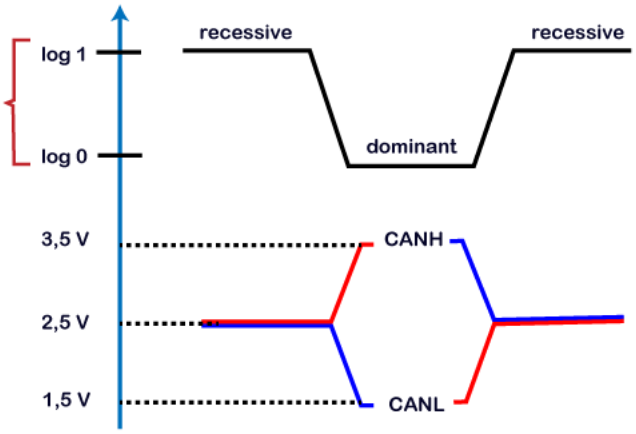
- **Step4:When Node1 transmit Node3 transmit Data continue to Node4**



5.8 CAN Structure



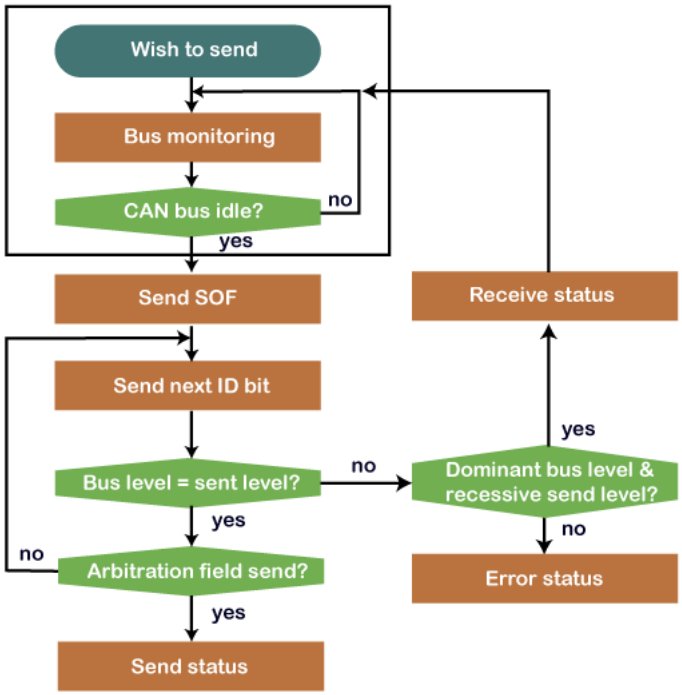
5.9 CAN Characteristics



Key points learnt from the CAN characteristics

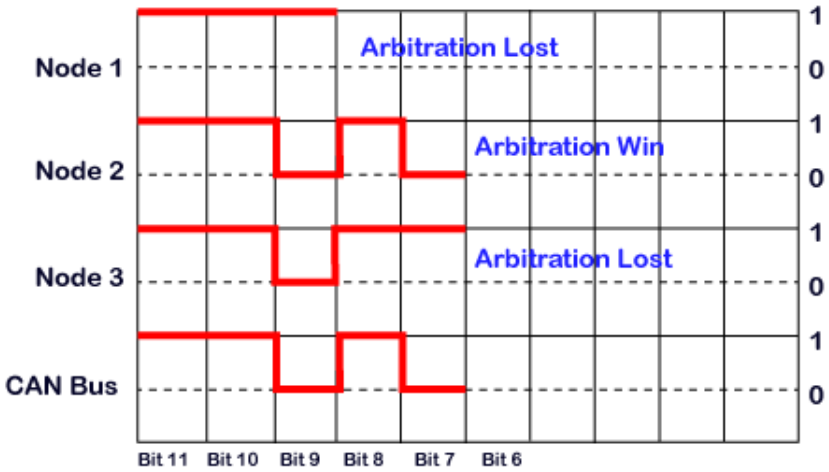
- Logic 1 is a recessive state. To transmit 1 on CAN bus, both CAN high and CAN low should be applied with 2.5V.
- Logic 0 is a dominant state. To transmit 0 on CAN bus, CAN high should be applied at 3.5V and CAN low should be applied at 1.5V.
- The ideal state of the bus is recessive.
- f the node reaches the dominant state, it cannot move back to the recessive state by any other node.

5.10 CAN Communication Principle



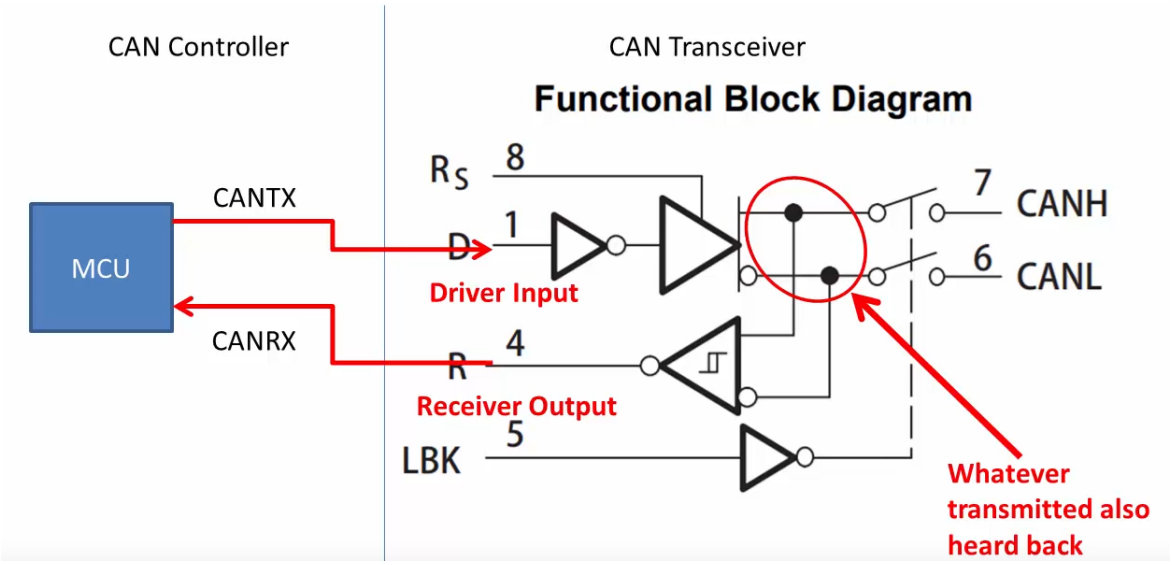
5.11 CAN Protocol Arbitration

CAN Node	Identifier (Hex)	Identifier (Binary)
1	0x7F3	11111110011
2	0x6B3	11010110011
3	0x6D9	11011011001



5.12 Controler area network with STM32 (bxCAN)

5.12.1 Transceiver function



5.12.2 Signaling Summary

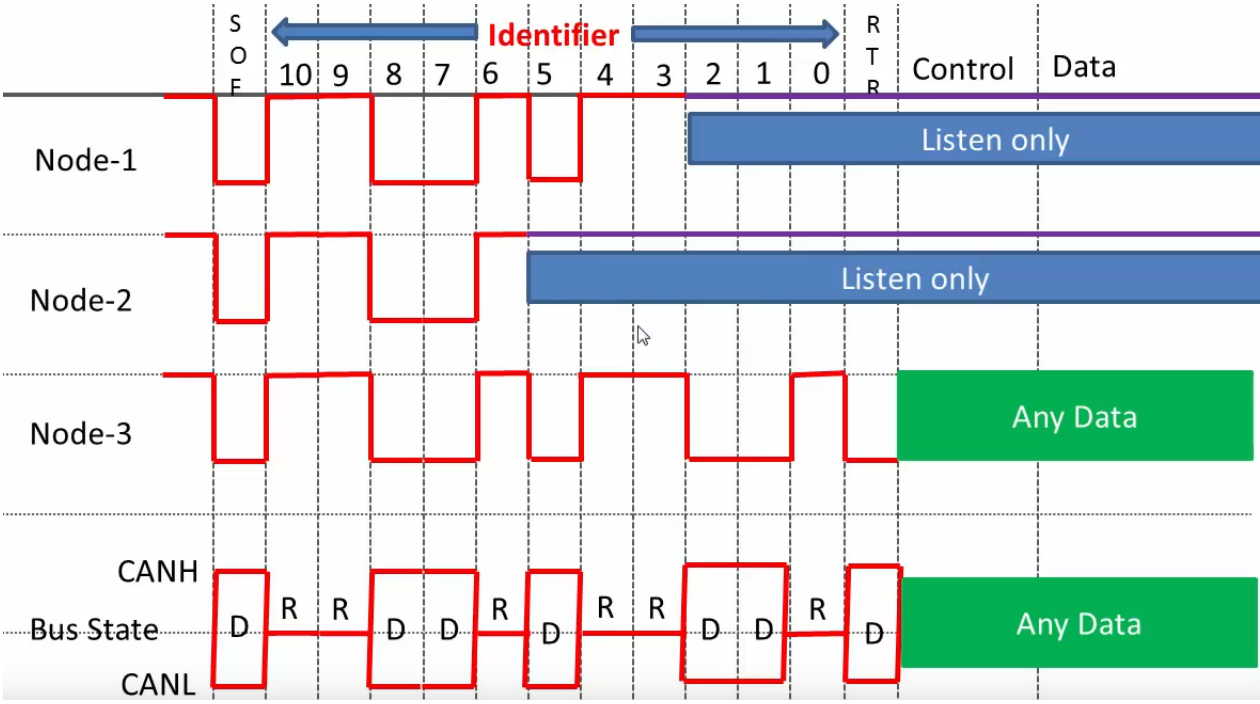
- Signalling is differential which is where CAN derives its robust noise immunity and fault tolerance.
- Balanced deifferential signalling reduces noise coupling and allows for high signalling rates over twisted-pair cable.
- Balanced means that the current flowing in each signal line is equal but opposite in direction ,resultng in a field-cancelling effect that is a key to low noise emissions.
- The use of balanced differential receivers and twisted-pair coupling enhance the common-mode rejection and high noise of a CAN_BUS immunity
- The cable is specified to be a shielded ot unshielded twisted-pair with a 120 ω characteristic impendence (Z0)

5.12.3 Bit wise bus arbitration animation _Bus access from Multiple Nodes

- The CAN communication protocol is a carrier-sense,multiple-access protocol with collision detection and arbitration on message priority (CAMA/CD+AMP)
- CSMA mean that each node on a bus must wait for a prescribed period of inactivity before attempting to send a message .CD+AMP means that collisions are resolved through a bit-wise arbitration ,bused on a preprogramed priority of each message in the identifier field of a message .
- The higher priority identifier always wins bus acecss.

Example:

Nodes	Arbitration ID
NODE-1	0x65D (110 0101 1101 b)
NODE-2	0x676 (110 0111 0110 b)
NODE-3	0x659 (110 0101 1001 b)



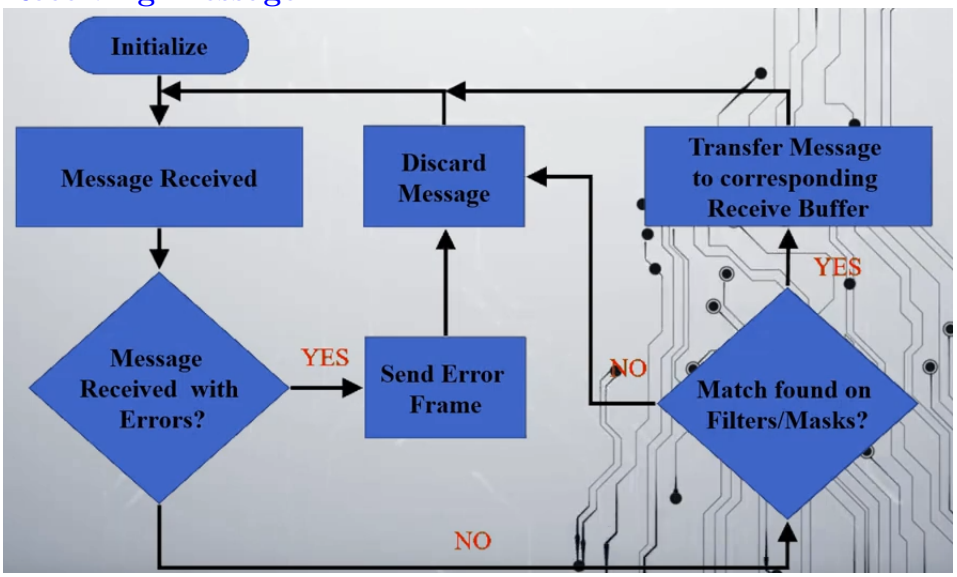
Why Node2 and Node3 stop?: Node2 and Node3 out of arbitration and enters listen only mode because it transmitted recessive state but finds bus state as dominant hence understands that it lost the arbitration.

5.12.4 CAN Message Formats

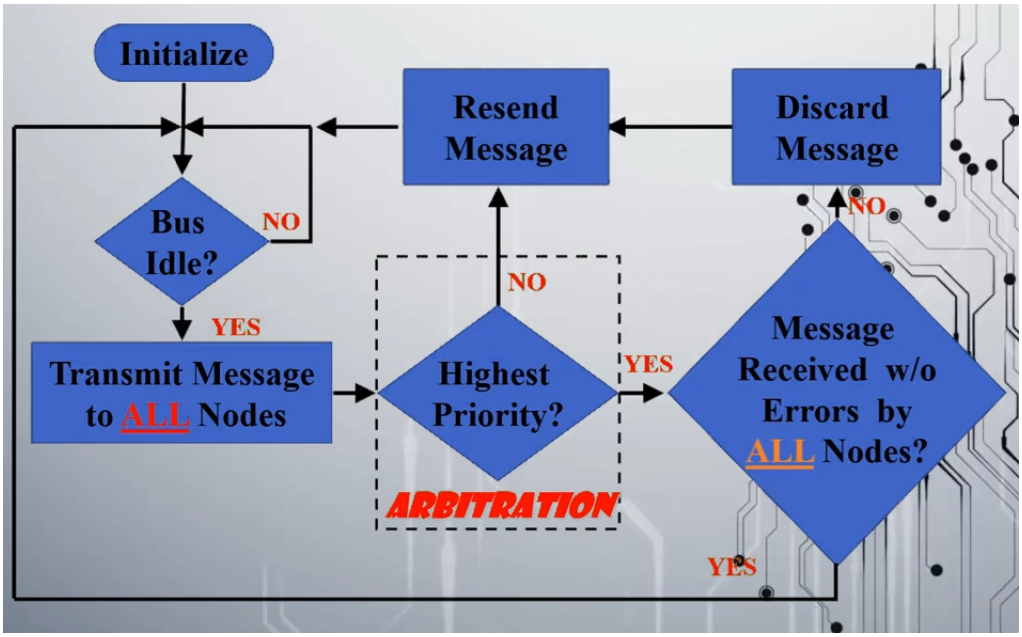
The CAN standard defines four message types

- **Data Frame:**This is used when a node transmits information to any or all other nodes in the system.
- **Remote Frame:** Similar to Data Frame without Data.
- **Error Frame:** a special message which is transmitted when a node detects a fault and will cause all other nodes to detect a fault.
- **Overload Frame:** similar to the Error Frame with regard to the format and it is transmitted by a node that becomes too busy.

Receiving Message



Transmit Message



5.12.5 Identifier filtering

In the CAN protocol the identifier of a message is not associated with the address of a node but related to the content of the message.On message reception a receiver node decides - depending on the identifier value - whether the software needs the message or not. If the message is needed, it is copied into the SRAM. If not, the message must be discarded without intervention by the software.

Scalable width

To optimize and adapt the filters to the application needs, each filter bank can be scaled independently. Depending on the filter scale a filter bank provides:

- One 32-bit filter for the STDID[10:0], EXTID[17:0], IDE and RTR bits.
- Two 16-bit filters for the STDID[10:0], RTR, IDE and EXTID[17:15] bits.

Mask mode

In mask mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”.

Identifier list mode

In identifier list mode, the mask registers are used as identifier registers. Thus instead of defining an identifier and a mask, two identifiers are specified, doubling the number of single identifiers. All bits of the incoming identifier must match the bits specified in the filter registers.

Filter bank scale and mode configuration

5.12.6 bxCAN Block Diagram (Tx-Path)

- Three transmit maiboxes are provided to the software for setting up messages.
- The transmission scheduler decides which mailbox has to be transmitted first.
- In order to transmit a message ,the application must select one empty transmit mailbox,setup the identifier,the data length code (DLC) and the Data before requesting the transmission.
- Request Transmission by setting TxRQ bit in the control register.
- Immediately aftter the TxRQ bit has been set,the mailbox enters pending state and waits to become the highest priority mailbox
- As soon as the mailbox has the highest priority it will be scheduled for transmission.

- The transmission of the message of the schedules mailbox will start (enter transmit state) when the CAN bus becomes idle
- Once the mailbox has been succesfully transmitted,it will becomes empty again.
- The hardware indicates a successful transmission by setting the RQCP and TxOK bits in the CAN_TSR register in case of an Arbitration Lost, and /or the TERR bit,in case of transmission error detection.
- If the transmission fails,the cause is indicated bt the ALST bit in the CAN_TSR register in case

5.12.7 Frame acception Rules:Example

- Accept frames only if first 3 msbs of the standard identifier are 1s e.g.111xxxx
- Accept frames only if first 3 msbs of the standard identifier are 0s ans last 2 lsbs are 1s
- Accept frames only if standard identifier value exaclty = 0x65D or 0x651
- Accept only Request frames
- Accept only Extended ID Frames
- Accept all Frames

5.12.8 bxCAN peripheral introduction

_ST's BxCAN Controller

The bxCAN(Basic xtended CAN)module handles the transmission and the reception of CAN message fully autonomously standard identifiers(11-bit) and extended identifiers (29-bits) are fully supported by the hardware.

_ST's BxCAN main features

- Supports CAN protocol version 2.0 A,B Active
- Bit rates up to 1Mbit/s
- Support the Time Triggered Communication option

Transmission

- Three transmit mailboxes
- configurable transmit priority
- Time Stamp on SOF transmission

Reception

- Two receive FIFOs with three stages
- Scalable filter banks:
  _ 28 filters banks shared between CAN1 and CAN2 in connectivity line devices
  _ 14 filters banks in other STM32F10xxx devices
- identifier list feature
- Configurable FIFO overrun
- Time Stamp on SOF reception

Timer-triggered communication option

- Disable automatic retransmission mode
- 16-bit free running timer
- Time Stamp sent in last two data bytes.

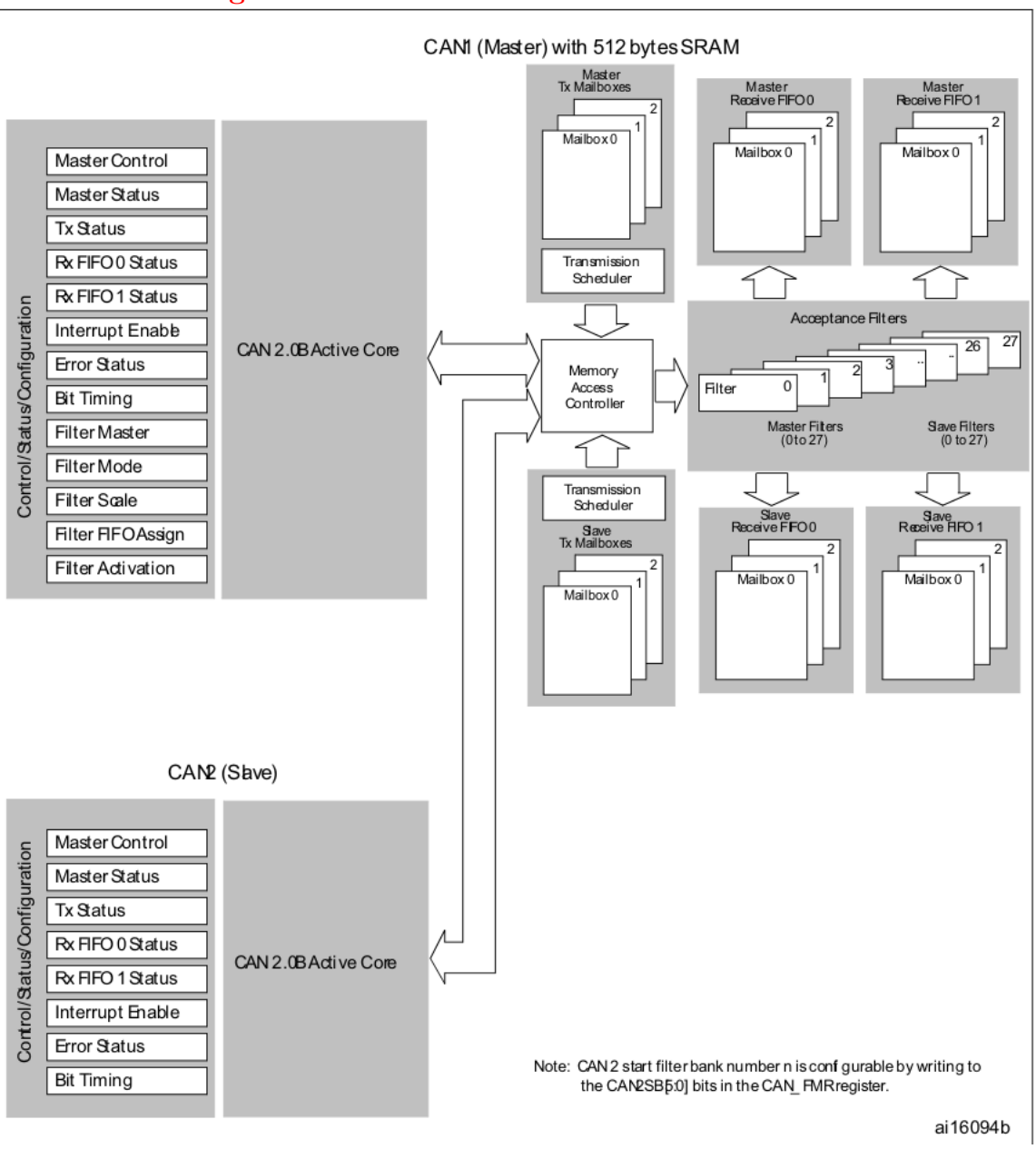
Management

- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

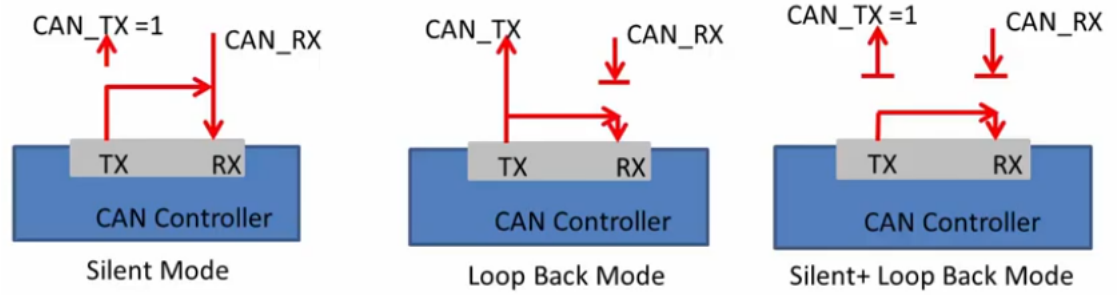
Dual CAN

- CAN1:Master bxCAN for managing the communication between a Slave bxCAN and the 512-byte SRAM memory
- CAN2:Slave bxCAN,with no direct access to the SRAM memory.

5.12.9 bxCAN block diagram



5.12.10 BxCAN Test Modes



Silent mode: In Silent mode, the bxCAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the bxCAN has to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Core

monitors this dominant bit, although the CAN bus may remain in recessive state. Silent mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames).

Loop back mode: This mode is provided for self-test functions. To be independent of external events, the CAN Core ignores acknowledge errors (no dominant bit sampled in the acknowledge slot of a data / remote frame) in Loop Back Mode. In this mode, the bxCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CANRX input pin is disregarded by the bxCAN. The transmitted messages can be monitored on the CANTX pin.

Loop back combined with silent mode:

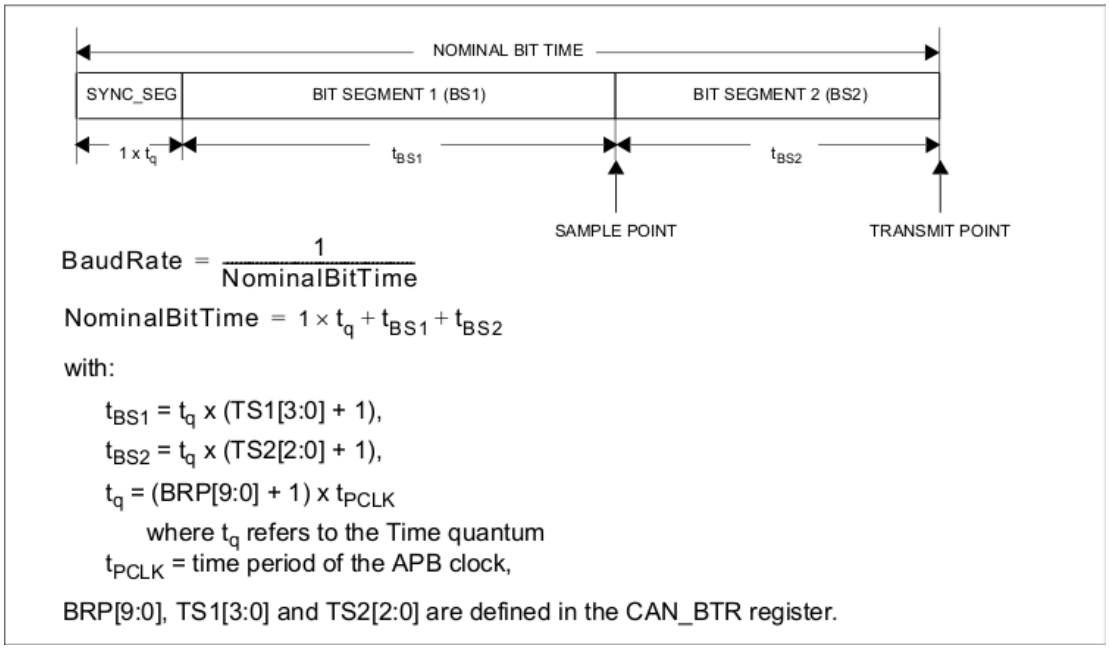
It is also possible to combine Loop Back mode and Silent mode by setting the LBKM and SILM bits in the CAN_BTR register. This mode can be used for a "Hot Selftest", meaning the bxCAN can be tested like in Loop Back mode but without affecting a running CAN system connected to the CANTX and CANRX pins. In this mode, the CANRX pin is disconnected from the bxCAN and the CANTX pin is held recessive.

5.12.11 Bit timing

The bit timing logic monitors the adjustment of the sample point by synchronizing on the star-bit edge and resynchronizing on the following edges on the following edges

Its operation may be explained simply by splitting nominal bit time into three segments as follows:

- **Synchronization segment(SYNC_SEG)**: a bit change is expected to occur within this time segment.it has a fixed length of one time quantum($1 \times t_q$).
- **Bit segment1 (BS1):** defines the location of the sample point.It includes the between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to difference in the frequency of the various nodes of the network.
- **Bit segment 2 (BS2)**:defines the location of the transmit point .it represents the PHASE_SEG2 of the CAN standard.Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.



VIII Component

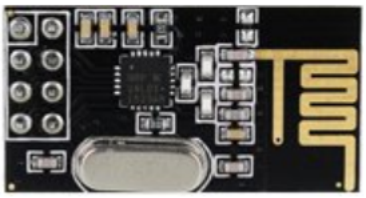
1. **NRF24L01**

1.1 Hardware Overview

- The nRF24L01+ module is designed to operate in the 2.4 GHz worldwide ISM frequency band and uses GFSK modulation for data transmission.
- The data transfer rate is configurable and can be set to 250kbps, 1Mbps, or 2Mbps.
- operating voltage ranges from 1.9 to 3.9V.
- The output power of the module can be programmed to be 0 dBm, -6 dBm, -12 dBm, or -18 dBm.
- At 0 dBm, the module consumes only 12 mA during transmission,
- 1.9 to 3.6V supply range
- Communication Range 800+ m (line of sight)
- Min. Current(Standby Mode) 26µA
- Max. Operating Current 13.5mA

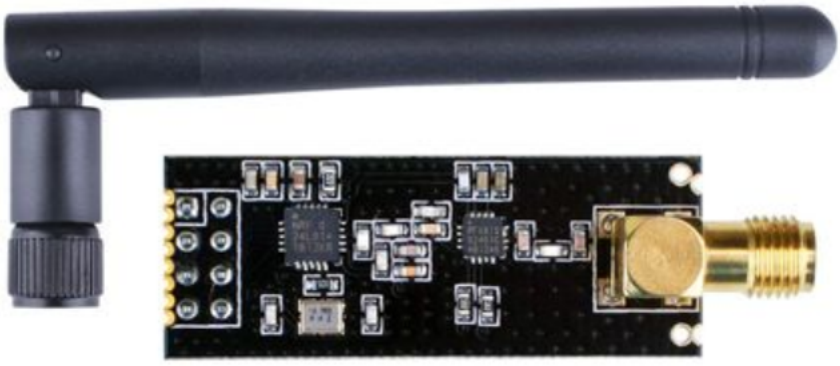
1.2 NRF24L01+ module -vs- NRF24L01+ PA/LNA module

- NRF24L01+ module:You will be able to communicate over a distance of 100 meters using this module. Of course, that is outside in the open. Its range gets a little weaker inside the house, especially because of the walls.



nRF24L01+ Wireless Module

- NRF24L01+ PA/LNA module:This enables the module to achieve a significantly greater transmission range of up to 1000 meters.



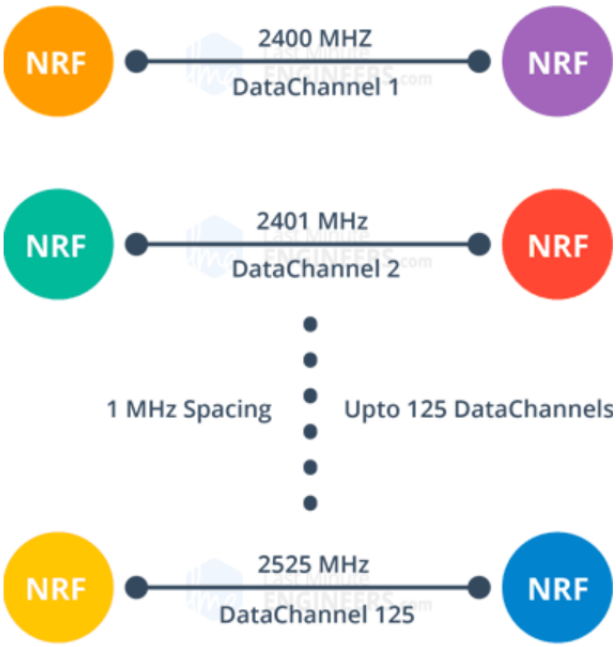
nRF24L01+ PA LNA Wireless Transceiver Module with External Antenna

- What exactly are PA and LNA?
  - PA stands for Power Amplifier.
  - LNA stands for Low-Noise Amplifier, and its function is to amplify an extremely weak signal received from the antenna (usually below microvolts or -100 dBm) to a more useful level (usually around 0.5 to 1 V).

1.3 How does the nRF24L01+ module work?

RF Channel Frequency

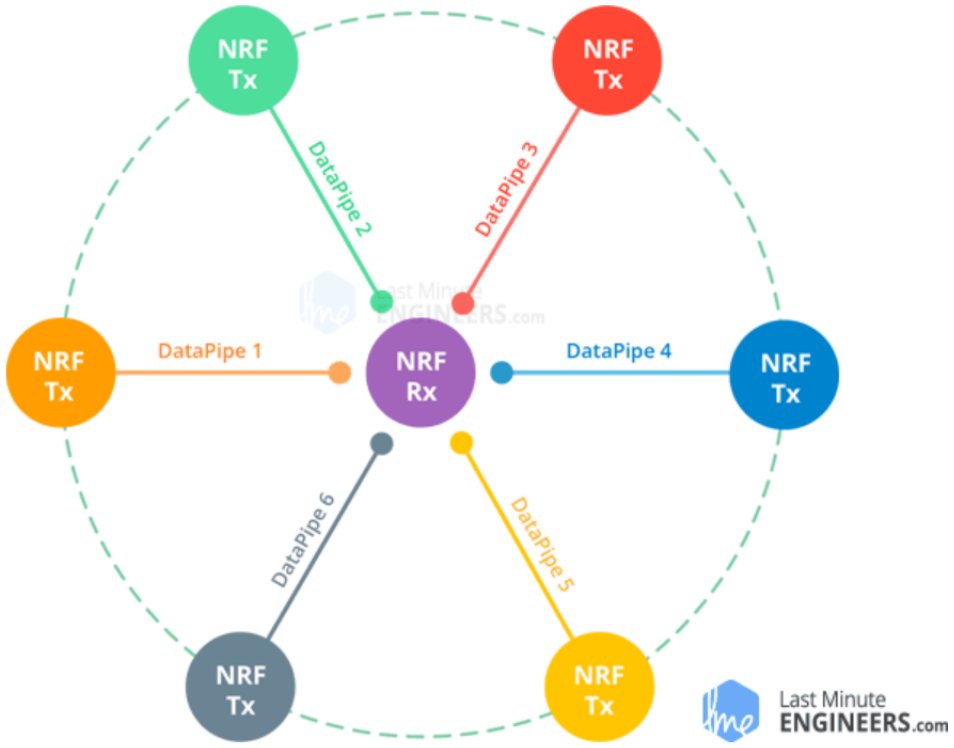
The nRF24L01+ module transmits and receives data on a specific frequency known as a channel. For two or more modules to communicate with each other, they must be on the same channel. This channel can have any frequency in the 2.4 GHz ISM band, or more precisely, any frequency between 2.400 and 2.525 GHz (2400 to 2525 MHz).
This means that the nRF24L01+ can operate on 125 different channels, allowing you to build a network of 125 independently operating modems in one location.



The RF channel frequency of your selected channel is calculated using the following formula:

$$Freq_{(Selected)} = 2400 + CH_{(Selected)}$$

For example, if you choose channel 108 for data transmission, the RF channel frequency will be 2508 MHz (2400 + 108).
NRF24L01+ Multiceiver Network
It stands for Multiple Transmitter Single Receiver.
A multiceiver network is depicted below.

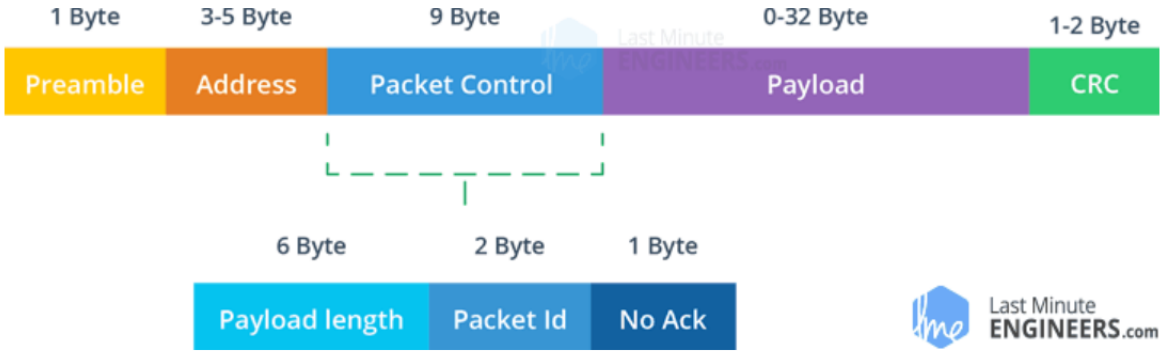


nRF24L01+ Multiceiver Network – Multiple Transmitters Single Receiver

To understand a multiceiver network, imagine the primary receiver acting as a hub receiver, collecting data from six different transmitter nodes at the same time. The hub receiver can switch from listening to transmitting at any time.

Enhanced ShockBurst Protocol

The nRF24L01+ uses a packet structure known as Enhanced ShockBurst. It has five fields: 0

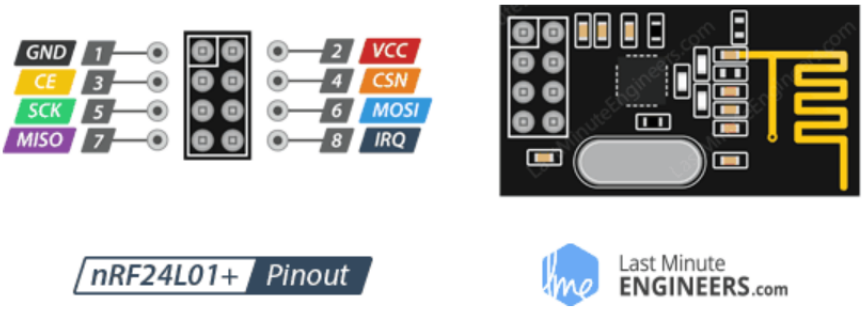


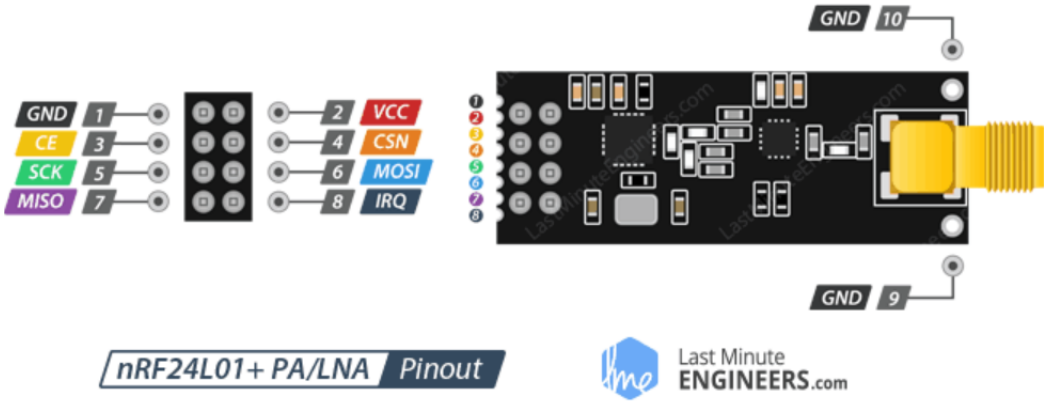
nRF24L01+ Enhanced ShockBurst Packet Structure

This new structure is excellent for a number of reasons.

- It supports variable length payloads with a payload length specifier, allowing payloads to range from 1 to 32 bytes.
- Each sent packet is assigned a packet ID, which allows the receiver to determine whether the message is new or has been retransmitted.
- Each message contains a field asking the receiver to send an acknowledgment.

NRF24L01+ Module Pinout





- CE (Chip Enable):is an active-high pin. When enabled, the nRF24L01 will either transmit or receive, depending on the mode.
- CSN (Chip Select Not):is an active-low pin that is typically held HIGH. When this pin goes low, the nRF24L01 begins listening for data on its SPI port and processes it accordingly.
- SCK (Serial Clock):accepts clock pulses from the SPI bus master.
- MOSI (Master Out Slave In):is the SPI input for the nRF24L01.
- MISO (Master In Slave Out):is the SPI output of the nRF24L01.
- IRQ:is an interrupt pin that can notify the master when there is new data to process.

2. **Laser_distance**

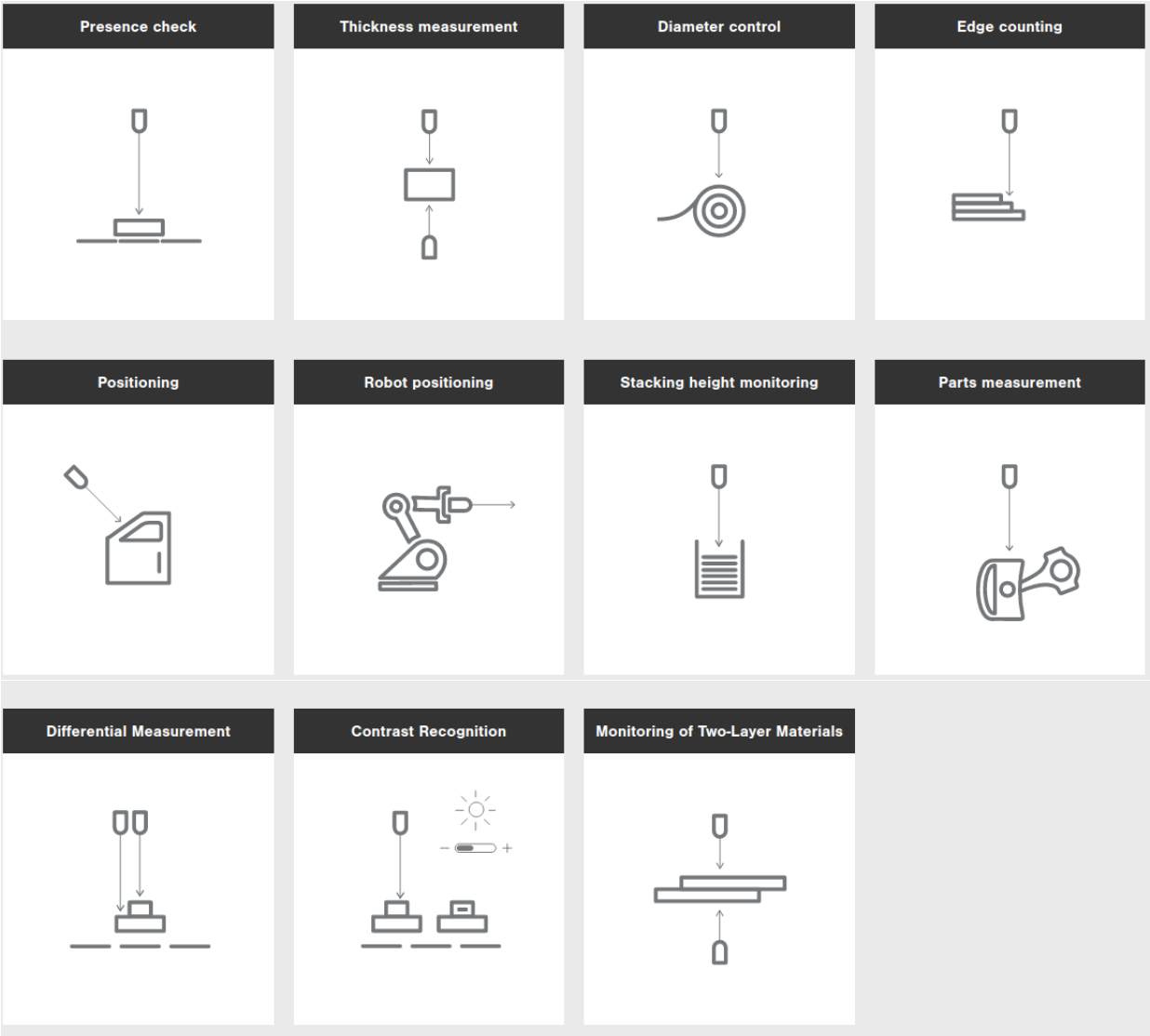
2.1 **What is Laser_distance?**

Laser distance measurer or laser distance meter measures distances between two points with speed and precision. The start point of the measurement is the set measurement edge of the laser distance meter and the second point is the end point of the beam.

2.2 **How Do Laser Distance Sensors Work?**

Laser sensors are photo-electronic sensors and, thanks to their contact less measuring principle and high accuracy, are well-suited to object detection, path, position and distance measurement. wenglor laser distance sensors work according to the transit time measurement principle and use laser triangulation. In both procedures, distances are measured with laser light and output as a distance value.

2.3 **Application**



2.4 Type of Laser distance

- 2-D LIDAR Sensor

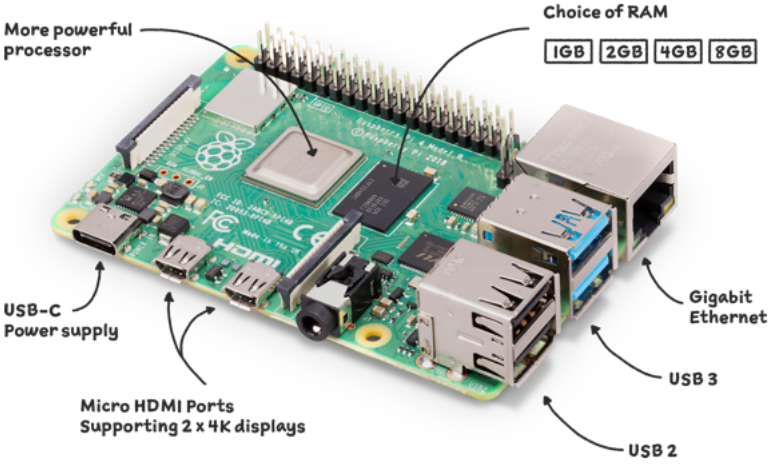
- 3-D LIDAR Sensor

- Light scanner
- Sensor with time of flight measurement
- Sensor with Laser triagulation

3. **Camera detect object**
4. **STM32**
5. **Raspberry pi**

5.1 **What is Raspberry pi?**

Raspberry Pi is defined as a minicomputer the size of a credit card that is inter operable with any input and output hardware device like a monitor, a television, a mouse, or a keyboard – effectively converting the set-up into a full-fledged PC at a low cost and to learn how to program in languages like Scratch and Python.



5.2 **How to work?**

Raspberry Pi is a programmable device. It comes with all the critical features of the motherboard in an average computer but without peripherals or internal storage. To set up the Raspberry computer, you will need an SD card inserted into the provided space. The SD card should have the operating system installed and is required for the computer to boot. Raspberry computers are compatible with Linux OS. This reduces the amount of memory needed and creates an environment for diversity.

After setting up the OS, one can connect Raspberry Pi to output devices like computer monitors or a High-Definition Multimedia Interface (HDMI) television. Input units like mice or keyboards should also be connected. This minicomputer's exact use and applications depend on the buyer and can cover many functions.

5.3 Application

- Media streamer
- Tablet computer
- Home automation
- Internet radio
- Robotics command
- Arcade devices
- cosmic computer
- Projects that use the Raspberry Pi

5.4 Top 6 Models of Raspberry Pi

- **Raspberry Pi Zero**
  This is the cheapest Raspberry model produced by the company. One can get it for as low as $5, which is quite impressive considering the extent of its functionality. Although not the first model to be released, it boasts a smaller, more compact size than the Raspberry Pi 1. Raspberry Pi Zero has the same processor and RAM (512 MB) as the Pi 1 Model B+. The Raspberry Pi Zero does not come with Wi-Fi or Bluetooth, but it can be made internet accessible via USB.
- **Raspberry Pi 1**
  Raspberry Pi 1 Model B was released in 2012. It served as a baseline in size for future releases. Initially, it had 26 GPIO pins, 256MB RAM capacity, and a single CPU core. You couldn't use it for heavy tasks with high processing needs. In 2014, the Raspberry Pi B+ was released with a starting RAM capacity of 512MB and 40 GPIO pins, becoming standard across all other models. Raspberry Pi Model B+ is sold at $25 and comes with four USB ports and an Ethernet connection. Pi 1 Model A+ ($20) can be considered for faster CPU processing speed, but it comes without an Ethernet connection.
- **Raspberry Pi 2 B**
  In February 2015, Raspberry released the 2B model. Compared to the prior releases, Raspberry Pi 2 B significantly improved, specifically in memory and speed. The RAM capacity was increased to 1GB. Pi 2B comes in the standard size, with 4 USB ports. It is currently priced at about $35, which is pretty affordable.
- **Raspberry Pi 3**
  Raspberry Pi 3 B was released in 2016. The B+ version, which came out in 2018, can boast a faster processing unit, Ethernet (802.11ac), and Wi-Fi than the earlier version. Generally, Raspberry PI 3 offers the user a wide range of use. It comes with the standard HDMI and USB ports, 1GB RAM, and Wi-Fi and Bluetooth connections in addition to the already functional Ethernet. One remarkable thing about this model is that it doesn't generate much heat or consume too much power. This makes it suitable for projects that require passive cooling and can be acquired at $35.
- **Raspberry Pi 4B**
  Released in 2019, Raspberry 4B is a vast improvement from its predecessors, with

a varying memory capacity from 2GB RAM to 8GB RAM. It also has a faster 1.5GHz processor and a good mix of 2.0 and 3.0 USB ports. Pi 4B is an ideal Raspberry model as it is suitable for virtually every use case with higher RAM capacity to satisfy even the most dedicated programmers. Depending on memory, the price ranges from $35 to $75, but each comes with all connectivity options.

- **Raspberry Pi 400**
  This model is unique as it comes in the form of a keyboard. It was launched in 2020 and operated with 4GB RAM. It comes with standard USB ports and needs just a monitor and mouse to make it a home computer set. Pi 400 costs $70 and can be used effectively in classrooms.

5.5 Top 10 Features of Raspberry Pi

- **Central Processing Unit(CPU)**
  Every computer has a Central Processing Unit, and so does the Raspberry Pi. It is the computer's brain and carries out instructions using logical and mathematical operations. Raspberry Pi makes use of the ARM11 series processor on its boards.
- **HDMI Port**
  Raspberry Pi board has an HDMI or High Definition Multimedia Interface port that allows the device to have video options of the output from the computer displayed. An HDMI cable connects the Raspberry Pi to an HDTV. The supported versions include 1.3 and 1.3. It also comes with an RCA port for other display options.
- **Graphic Processing Unit(GPU)**
  This unit, GPU or Graphic Processing Unit, is another part of the Raspberry pi board. Its primary purpose is to hasten the speed of image calculations.
- **Memory(RAM)**
  Random Access Memory is a core part of a computer's processing system. It is where real-time information is stored for easy access. The initial Raspberry Pi had 256MB RAM. Over the years, developers gradually and significantly improved the size. Different Raspberry Pi models come with varying capacities. The model with the maximum capacity presently is the Raspberry Pi 4 with 8GB RAM space.
- **Ethernet Port**
  The Ethernet port is a connectivity hardware feature available on B models of Raspberry Pi. The Ethernet port enables wired internet access to the minicomputer. Without it, software updates, web surfing, etc., would not be possible using the Raspberry Pi. The Ethernet port found on Raspberry computers uses the RJ45 Ethernet jack. With this component, Raspberry Pi can connect to routers and other devices.
- **SD card slot**
  Like most other regular computers, Raspberry Pi must have some sort of storage device. However, unlike conventional PCs, it does not come with a hard drive, nor does it come with a memory card. The Raspberry Pi board has a Secure Digital card or SD card slot where users must insert SD cards for the computer to function. The SD card functions like a hard drive as it contains the operating system necessary for turning the system on. It also serves to store data.
- **General Purpose Input and Output (GPIO) pins**
  These are upward projecting pins in a cluster on one side of the board. The oldest models of the Raspberry Pi had 26 pins, but most have 40 GPIO pins. These pins are pretty sensitive and should be handled carefully. They are essential parts of the Raspberry Pi device as they add to its diverse applications. GPIO pins

are used to interact with other electronic circuits. They can read and control the electric signals from other boards or devices based on how the user programs them.

- **LEDs**
  These are a group of five light-emitting diodes. They signal the user on the present status of the Raspberry Pi unit. Their function covers:
  _PWR (Red): This functions solely to indicate power status. When the unit is on, it emits a red light and only goes off when the unit is switched off, or disconnected from the power source.
  _ACT (Green): This flashes to indicate any form of SD card activity.
  _LNK (Orange): LNK LED gives off an orange light to signify that active Ethernet connectivity has been established.
  _100 (Orange): This light comes on during Ethernet connection when the data speed reaches 100Mbps.
  _FDX (Orange): FDX light also comes during Ethernet connection. It shows that the connection is a full-duplex.
- **USB ports**
  Universal service bus (USB) ports are a principal part of Raspberry Pi. They allow the computer to connect to a keyboard, mouse, hard drives, etc. The first model of Raspberry Pi had only two USB 2.0 ports. Subsequent models increased this number to four. Raspberry Pi 4 and Pi 400, much newer models, come with a mix of USB 2.0 and USB 3.0 ports.
- **Power source**
  Raspberry Pi has a power source connector that typically uses a 5V micro USB power cable. The amount of electricity any Raspberry Pi consumes depends on what it's used for and the number of peripheral hardware devices connected.

5.6 Advantages

- The Raspberry Pi is a small computer that is roughly the size of a credit card.
- The Raspberry Pi is inexpensive.
- Using a group of Raspberry Pi's to function as a server is more efficient than using a regular server.
- The Raspberry Pi is ideal for adaptive technology because it can display visuals and play movies.
- This microcomputer can be used by small businesses on a tight budget to use their product or build new technology that integrates the product.

5.7 Disadvantages

- It is not a computer replacement, and the processor is not as fast. Downloading and installing software takes time so that you won't do any intricate multitasking.
- Other operating systems, such as Windows, are incompatible.
- If the extra work is worth it, business owners should think about it.
- This product will not be beneficial for larger businesses that already have large servers, which can perform all of the Raspberry Pi tasks. As a result, it would not be worth it, and putting everything together would take time.

6. NVIDIA Jetson

6.1 What Is NVIDIA Jetson?

NVIDIA Jetson provides advanced embedding systems that enable you to create artificial intelligence (AI) products for various scenarios. It is a power-efficient hardware platform for AI that consists of modular, high-performance, small-form-factor edge

computers. NVIDIA Jetson also offers the JetPack SDK for software acceleration and an ecosystem for speeding up the development of custom AI projects.

6.2 What Are the Benefits of the Jetson Platform?

- NVIDIA Jetson is suitable for organizations of all sizes, and also for students or individual developers. It provides a set of modules that can be useful for anything from entry-level AI applications to highly complex AI-powered devices.
- NVIDIA Jetson is powered by a unified software architecture that frees developers from the hassle of repetitive coding. Whenever they require AI/ML capability, they can add a relevant Jetson module to the device and it takes care of the heavy lifting.
- The NVIDIA JetPack SDK comes with a Linux operating system, CUDA-X acceleration libraries, and APIs for various machine learning domains, including deep learning and computer vision. It also supports machine learning frameworks such as TensorFlow, Caffe and Keras, and computer vision libraries such as OpenCV.
- The NVIDIA Jetson platform supports cloud-native technologies and workflows such as containerization and orchestration, giving developers the agility to rapidly develop and scale up AI products.

6.3 NVIDIA Jetson Modules

- **Jetson AGX Orin Developer Kit**
  This kit allows developers to build a full-featured AI application using Jetson Orin modules. It provides a power-efficient, high-performance Jetson AGX Orin module and can emulate other modules.

Component	Specifications
GPU	NVIDIA Ampere architecture with 2048 NVIDIA® CUDA® cores and 64 Tensor cores
CPU	12-core Arm Cortex-A78AE v8.2 64-bit CPU 3MB L2 + 6MB L3
DL Accelerator	2x NVDLA v2.0
Vision Accelerator	PVA v2.0
Memory	32GB 256-bit LPDDR5 204.8 GB/s
Storage	64GB eMMC 5.1

- **Jetson AGX Xavier**
  Jetson AGX Xavier is a new module that lets you build an AI-powered autonomous machine. It runs on just 10W and delivers up to 32 TOPs. AGX Xavier benefits from the NVIDIA ecosystem, including a rich selection of AI workflows and tools provided by the major AI platform. Developers can use these offerings to train and deploy a neural network quickly.

Component	Specifications
AI Performance	32 TOPs
GPU	512-core Volta GPU with 64 Tensor Cores 11 TFLOPS (FP16) 22 TOPS (INT8)
CPU	8-core Carmel ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
Memory	32GB 256-Bit LPDDR4x | 136.5GB/s
Storage	32GB eMMC 5.1
DL Accelerator	(2x) NVDLA Engines, 5 TFLOPS (FP16), 10 TOPS (INT8)
Vision Accelerator	7-way VLIW Vision Processor

- **Jetson AGX Xavier Industrial**
  This module is part of the NVIDIA Jetson AGX Xavier series. It has a pin-compatible form factor design, allowing you to leverage the most modern AI models for demanding applications. It offers extended shock, vibration, and temperature specifications, advanced security capabilities and features, and up to 20 times the performance and four times the memory of the NVIDIA Jetson TX2i module.

Component	Specifications
GPU	NVIDIA Volta™ architecture with 512 NVIDIA® CUDA® cores and 64 Tensor cores 20 TOPS (INT8)
CPU	8-core NVIDIA Carmel Arm®v8.2 64-bit CPU 8MB L2 + 4MB L3
DL Accelerator	2x NVDLA 10 TOPS (INT8)
Vision Accelerator	2x 7-Way VLIW Vision Processor
Memory	32GB 256-bit LPDDR4x (ECC support) 136.5GB/s
Storage	64GB eMMC 5.1

- **Jetson AGX Xavier Developer Kit**
  This developer kit lets you easily build and deploy an end-to-end AI-powered application for various robotics use cases, including manufacturing, retail, and delivery. It has wide support from NVIDIA JetPack, DeepStream SDKs, cuDNN, CUDA, and TensorRT, providing all the necessary tools to start an AI development project.

Component	Specifications
GPU	512-core Volta GPU with Tensor Cores
CPU	8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
Memory	32GB 256-Bit LPDDR4x | 137GB/s
Storage	32GB eMMC 5.1
DL Accelerator	(2x) NVDLA Engines
Vision Accelerator	7-way VLIW Vision Processor

- **Jetson Xavier NX**
  This module can deliver 21 TOPS to run a modern AI workload, consuming just 10 watts of power. Its form factor is more compact than a credit card. Xavier NX can run multiple neural networks simultaneously, processing data from various high-resolution sensors. It allows you to build applications for edge computing and embedded devices requiring high performance but with significant power, weight, and size constraints.

Component	Specifications
AI Performance	21 TOPS (INT8)
GPU	384-core NVIDIA Volta™ GPU with 48 Tensor Cores
CPU	6-core NVIDIA Carmel ARM®v8.2 64-bit CPU 6MB L2 + 4MB L3
Memory	8 GB 128-bit LPDDR4x @ 1866MHz 59.7GB/s
Storage	16 GB eMMC 5.1
DL Accelerator	2x NVDLA Engines
Vision Accelerator	7-Way VLIW Vision Processor

- **Jetson Xavier NX Developer Kit**
  This developer kit contains a compact power-efficient Xavier NX module for AI-powered edge devices. It has cloud native support (a new feature) and can accelerate the NVIDIA software stack in just 10 W with over ten times the performance of the popular Jetson TX2.

Component	Specifications
GPU	NVIDIA Volta architecture with 384 NVIDIA CUDA® cores and 48 Tensor cores
CPU	6-core NVIDIA Carmel ARM®v8.2 64-bit CPU 6 MB L2 + 4 MB L3
DL Accelerator	2x NVDLA Engines
Vision Accelerator	7-Way VLIW Vision Processor
Memory	8 GB 128-bit LPDDR4x @ 51.2GB/s
Storage	microSD (not included)

- **Jetson TX2 Module**
  This module is the fastest embedded-AI computing appliance offering high power efficiency (just 7.5 W). It offers supercomputer capabilities for edge AI devices. Built on a GPU from the NVIDIA Pascal family, it has 8GB of memory and a 59.7GB-per-second bandwidth. It offers various hardware interfaces making it easier to integrate into different form factors and products.

Component	Specifications
GPU	256-core NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores
CPU	Dual-Core NVIDIA Denver 2 64-Bit CPU Quad-Core ARM® Cortex®-A57 MPCore
Memory	8GB 128-bit LPDDR4 Memory 1866 MHx—59.7 GB/s
Storage	32GB eMMC 5.1
Power	7.5W / 15W

- **Jetson Nano**
  This model is the smallest machine for AI-embedded applications and IoT devices. It is extremely powerful for its size, delivering the power required for advanced AI projects in a $99 module. NVIDIA Jetson Nano Get lets you get started quickly using a comprehensive JetPack SDK and accelerated libraries for computer vision, deep learning, graphics, multimedia, and other applications. It offers the functionality and performance required to run a modern AI workload, letting you easily add AI capabilities to your products.

Component	Specifications
GPU	NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores
CPU	Quad-core ARM Cortex-A57 MPCore processor
Memory	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Storage	16 GB eMMC 5.1

7. **PS4**

IX Mathematical

1. **Linear Algebra**

1.1 MATRIX

1.1.1 Operation on matrices

- The matrix A is said to be orthogonal if $A^tA = I$
- $(A + B)^t = A^t + B^t$
- $(A^t)^t = A$
- $(\lambda A)^t = A$
- $(AB)^t = B^tA^t$
- $(E_{ij})^t = E_{i,j}$

1.1.2 Trace of a Matrix

- Let $A = (a_{ij})_n$. The trace of the matrix A is defined by:

$$tr(A) = \sum_{i=1}^{n} a_{ii} = a_{11} + a_{22} + ... + a_n$$

- $tr(A + B) = tr(A) + tr(B)$
- $tr(\lambda A) = \lambda tr(A)$
- tr(AB)=tr(BA)

1.1.3 System of linear Equations

Form : AX=b.

Wher

$$A \begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ \vdots & \vdots & ... & \vdots \\ a_{m1} & a_{m2} & ... & a_{mn} \end{bmatrix} X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} andb = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

1.1.4 Polynomial of a Matrix

Let A be a square matrix of order n. We form the power of matrix A as follow:

$$p(x) = a_0I + a_1A + a_2A^2 + ... + a_kA^k.$$

If a square matrix A such that p(A)=0, then the matrix A is called a zero of the polynomial p(x)

1.2 DETERMINANT

1.2.1 Permutations

- Let $\sigma = j_1 j_2 ... j_n$ of S_n. We say that $(\sigma(i), \sigma(k)) = (j_i, j_k)$ is an inversion of σ if

$$i < k \quad and \quad j_i > j_k \quad (\sigma(i) > \sigma(k))$$

- A permutation is called even or odd according to whether the total number of inversions in it is even or odd.
- The signature of a permutation σ, denoted sgn(σ), is defined as

$$sgn(\sigma) = (-1)^{\#inversions} = \begin{cases} 1, & if \sigma is even \\ -1, & if \sigma is odd \end{cases}$$

Example: Find the signatures of the following permutation.
σ=45312
you count from number big to small and conut from left to right
Such that: we get (4,3),(5,3) — (4,1),(5,1),(3,1) — (4,2),(5,2),(3,2)
Inversion=0+0+1+3+3=8
$\Longrightarrow$ σ is even permutation
$\Longrightarrow$ $sgn(\sigma) = (-1)^8$=+1

1.2.2 Determinants

Let A=$(a_{ij})_n$. the determinant of A (written det(A) of —A—) is defined by:

$$|A| = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \sum_{\sigma \epsilon S_n} sgn(\sigma) a_{1\sigma(1)} a_{2\sigma(2)} ... a_{n\sigma(n)} = \sum_{\sigma \epsilon S_n} sgn(\sigma) a_{1j_1} a_{2j_2} ... a_{nj_n}.$$

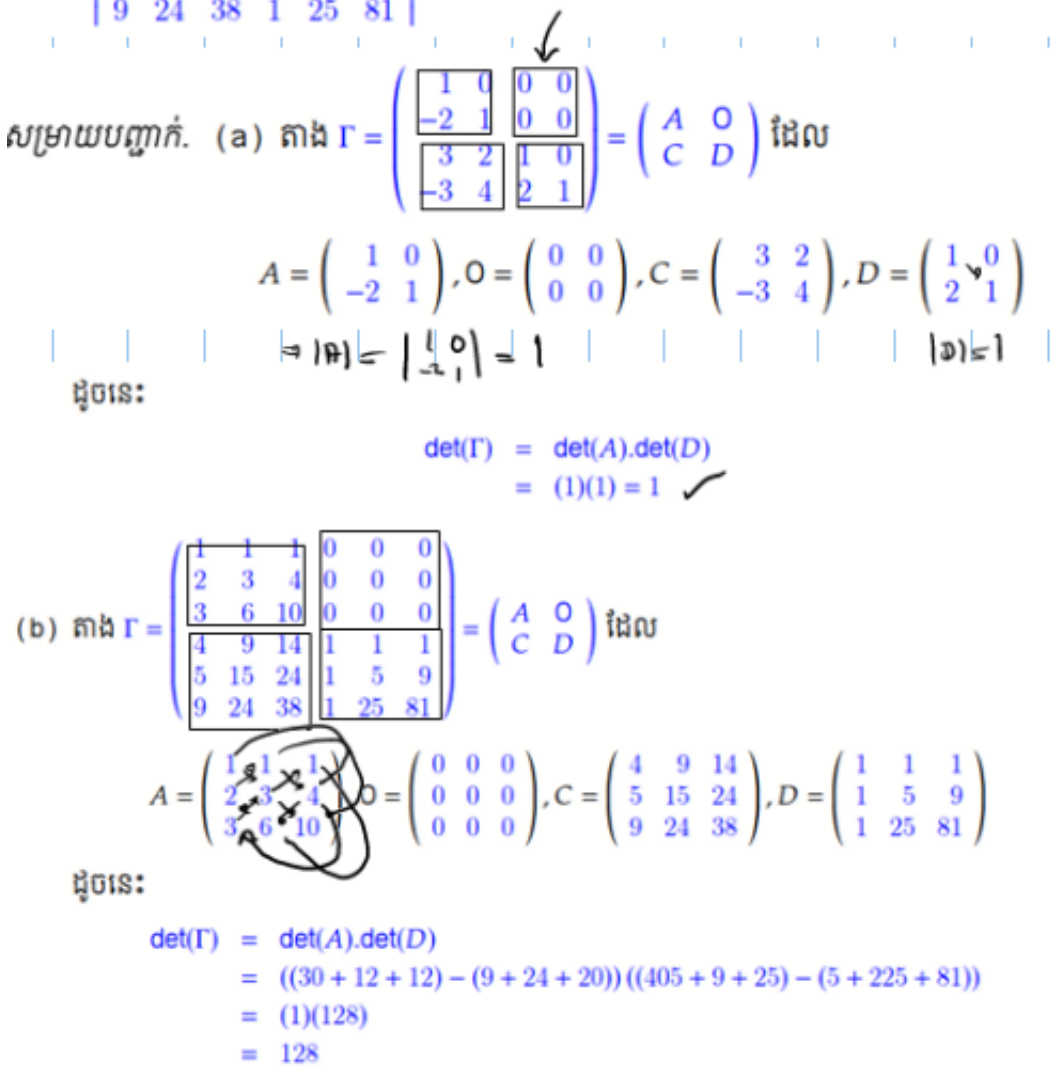
1.2.3 Properties of Determinants

- $|A^t| = |A|$
- $|AB| = |A|.|B|$
- $|A^{-1}| = \frac{1}{|A|}$
- $|\lambda A| = \lambda^n |A|$
- $\det\begin{pmatrix} A & O \\ C & D \end{pmatrix} = det\begin{pmatrix} A & B \\ O & D \end{pmatrix} = det(A)det(D)$
- $\det\begin{pmatrix} A & B \\ C & D \end{pmatrix}$ =det(AD-BC)

Example:

(a) $\begin{vmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ -3 & 4 & 2 & 1 \end{vmatrix}$

(b) $\begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 3 & 4 & 0 & 0 & 0 \\ 3 & 6 & 10 & 0 & 0 & 0 \\ 4 & 9 & 14 & 1 & 1 & 1 \\ 5 & 15 & 24 & 1 & 5 & 9 \\ 9 & 24 & 38 & 1 & 25 & 81 \end{vmatrix}$



សម្រាយបញ្ជាក់. (a) តាង $\Gamma = \begin{pmatrix} \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 3 & 2 \\ -3 & 4 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \end{pmatrix} = \begin{pmatrix} A & O \\ C & D \end{pmatrix}$ ដែល

$$A = \begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}, O = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, C = \begin{pmatrix} 3 & 2 \\ -3 & 4 \end{pmatrix}, D = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

ដូចនេះ:

$$\begin{aligned} \det(\Gamma) &= \det(A).\det(D) \\ &= (1)(1) = 1 \end{aligned}$$

(b) តាង $\Gamma = \begin{pmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 4 \\ 3 & 6 & 10 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 4 & 9 & 14 \\ 5 & 15 & 24 \\ 9 & 24 & 38 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 1 \\ 1 & 5 & 9 \\ 1 & 25 & 81 \end{bmatrix} \end{pmatrix} = \begin{pmatrix} A & O \\ C & D \end{pmatrix}$ ដែល

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 4 \\ 3 & 6 & 10 \end{pmatrix}, O = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, C = \begin{pmatrix} 4 & 9 & 14 \\ 5 & 15 & 24 \\ 9 & 24 & 38 \end{pmatrix}, D = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 5 & 9 \\ 1 & 25 & 81 \end{pmatrix}$$

ដូចនេះ:

$$\begin{aligned} \det(\Gamma) &= \det(A).\det(D) \\ &= ((30 + 12 + 12) - (9 + 24 + 20))((405 + 9 + 25) - (5 + 225 + 81)) \\ &= (1)(128) \\ &= 128 \end{aligned}$$

1.2.4 Determinant of Block Matrix

Let A, B, C , DϵM_n(R) with D is invertible. We have

$$det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = det(D) \times det(A - BD^{-1}C)$$

Let A,B,C,D $\epsilon M_n(R)$ with A is invertble.We have

$$det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = det(A) \times det(D - CA^{-1}B)$$

1.2.5 Minor and Cofactor

Let A $= (a_{ij})_n$. The adjoint of A is denoted and defind by:

$$adj(A) = (A_{ij})^t = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{pmatrix}$$

Let A be a square matrix.Then

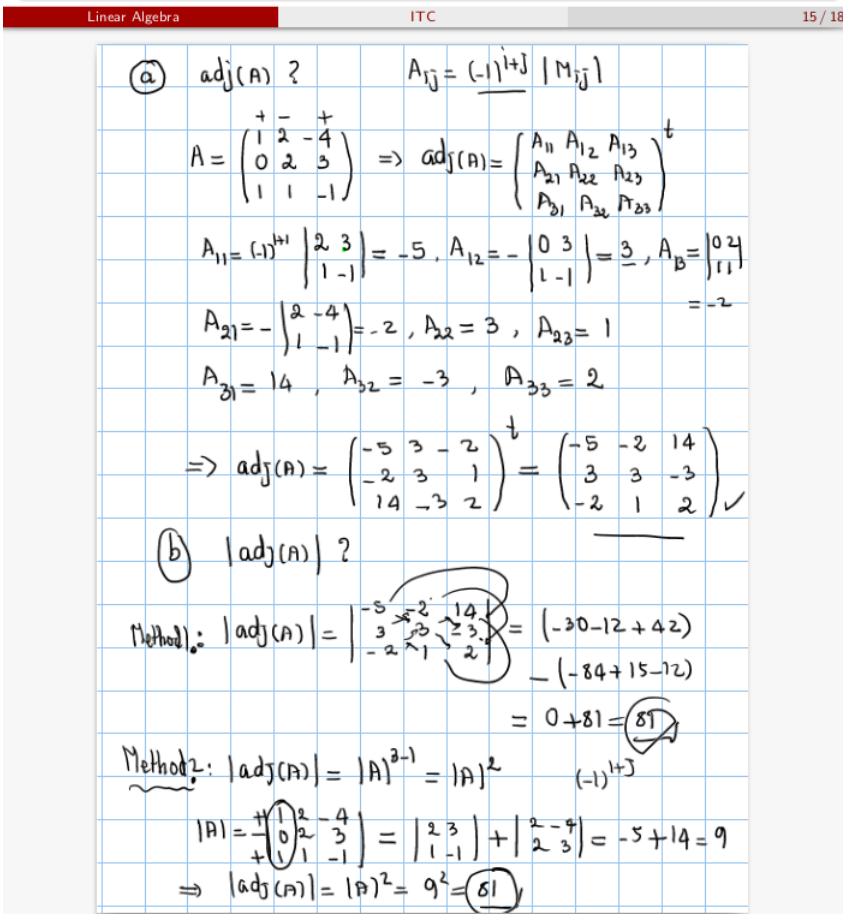
$$A.adj(A) = (adj(A))A = |A|I.$$

If $|A| \neq 0$, then $A^{-1} = \frac{1}{|A|}(adj(A))$
Let A,B $\epsilon M_n(R)$ ne invertible matrices and $\alpha \neq 0$.Then

- $adj(A^{-1} = (adj(A))^{-1}$
- $adj(A^t) = (adj(A))^t$
- $adj(AB) = adj(B)adj(A)$.
- $|adj(A)| = |A|^{n-1}$
- $adj(\alpha A) = \alpha^{n-1} adj(A)$.

Let $A = \begin{pmatrix} 1 & 2 & -4 \\ 0 & 2 & 3 \\ 1 & 1 & -1 \end{pmatrix}$.

Find
(a) adj(A), (b) |adj(A)|, (c) A^{-1}, (d) |adj(3A)|, (e) $|\text{adj}(2A^{-1})|$



1.2.6 Cramer's rule

Considers a system of n linear equation in the n unknowns:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

This system is equivalent to the matrix form

$$AX = b$$

The above system has a unique solution if and only if $\triangle \neq 0$.In this case,the unique solution is given by:

$$x_1 = \frac{\triangle_1}{\triangle}, x_2\frac{\triangle_2}{\triangle}, ..., x_n = \frac{\triangle_n}{\triangle}$$

Where :

- $\triangle = |A|$.
- $\triangle_i$,i=1,2,...,n, be the determinant of the matrix obtained by replacing the ith column of A by b.

Example1:

Example2:

Ex 21:

ឧទាហរណ៍ ២.៤២. គេមាន $a,b,c \in \mathbb{C}$ និងពហុធា $P(X) = X^3 - (x + yX + zX^2)$។ ដោះស្រាយប្រព័ន្ធសមីការខាងក្រោមដោយប្រើពហុធា P ។

$P(a) = a^3 - (x + ay + a^2z) =$

$P(a) = 0 \Leftrightarrow$ ——— $= 0$

$x + ay + az^2 = a^3$

(a)
$$\begin{cases} x + ay + a^2z &= a^3 \\ x + by + b^2z &= b^3 \\ x + cy + c^2z &= c^3 \end{cases}$$

(b)
$$\begin{cases} x + ay + a^2z &= a^4 \\ x + by + b^2z &= b^4 \\ x + cy + c^2z &= c^4 \end{cases}$$

សម្រាយបញ្ជាក់. (a) បើ x, y, z ជាចម្លើយនៃប្រព័ន្ធសមីការ នោះ $P(a) = P(b) = P(c) = 0$ ។ ដូចនេះ:

$$P(X) = (X-a)(X-b)(X-c)$$

$$= (X^2 - (a+b)X + ab)(x - c)$$
$$= X^3 - cX^2 - (a+b)X^2 + (ac+bc)X + abX - abc$$
$$= X^3 - (a+b+c)X^2 + (ab+bc+ca)X - abc$$

ដោយ $P(X) = X^3 - zX^2 - yX - x$ នោះយើងទាញបាន

$$x = abc, y = -(ab+bc+ca), z = a+b+c \checkmark$$

(b)
$$\begin{cases} x + ay + a^2z &= a^4 \\ x + by + b^2z &= b^4 \\ x + cy + c^2z &= c^4 \end{cases}$$

តាង $P(X) = X^4 - (x + yX + zX^2)$ នោះ $P(a) = P(b) = P(c) = 0$ យើងបាន

$$P(X) = (X-a)(X-b)(X-c)(X-d) \checkmark$$

ដែល d ជាឫសមួយទៀតនៃ $P(X)$ ។ ដូចនេះ យើងអាចសរសេរ

$$P(X) = (X^2 - (a+b)X + ab)(X^2 - (c+d)X + cd)$$
$$= X^4 - (c+d)X^3 + cdX^2 - (a+b)X^3 + (a+b)(c+d)X^2 - cd(a+b)X + abX^2 - ab(c+d)X$$
$$= X^4 - (a+b+c+d)X^3 + (ab+ac+ad+bc+bd+cd)X^2 - (abc+abd+acd+bcd)X +$$

$ab+ac+bc-(a+b+c)^2$

ដោយ $P(X) = X^4 - zX^2 - yX - x$ នោះយើងទាញបាន

$$a+b+c+d = 0 \Rightarrow d = -(a+b+c) \checkmark$$
$$x = -abcd = abc(a+b+c)$$
$$y = abc - (ab+ac+bc)(a+b+c) \checkmark$$
$$z = -[ab+ac+bc-(a+b+c)^2]$$

1.3 Vector Spaces

2.1 JORDAN CANONICAL FORM

2.1.1 Eigenvalues and Eigenvector

- The dimension of this eigenspace is called geometric multiplicity of λ denoted $gm(\lambda)$.That is,
$$gm(\lambda) = dim(E_\lambda)$$
- Let A $\epsilon M_n(K) and \lambda \epsilon K$.The following statements are equivalent.
_λ is an eigenvalue of A.
_(A-λI)x=0 has a montrivial solution.
_The eigenspace $E_\lambda \neq \{0\}$.
_det(A-λI)=0.

2.1.2 Polynomial

- Let V ϵV_n^K,L $\epsilon L(V)$,B be a basis for V, and A=$[L]_B$.Then

$$p_A(\lambda) = p_l(\lambda) = (-1)^n(\lambda^n - S_1\lambda^{n-1} + S_2\lambda^{n-2} + ... + (-1)^n S_n).$$

  Where S_i is the sum of the principle minors of order i of matrix A.
- Let V ϵV_n^K and L $\epsilon L(V)$. Let λ be an eigenvalue of L.Then

$$1 \leq gm(\lambda) \leq am(\lambda)$$

  .
- Let$p_A(\lambda)$ be the characteristic polynomial and m_A be the minimal polynomial of A.
  _The polynomial $m_A(\lambda)$ divides every polynomial that has A as a zero.
  _The polynomial $p_A(\lambda)$ and $m_A(\lambda)$ have the same irreducible factors over $\mathbb{k}$.
  _If $p_A(\lambda) = (\lambda_1 - \lambda)^{n_1}...(\lambda_K - \lambda)^{n_k}$ where $\lambda_1, ..., \lambda_k$ are distinct,then there exist integers $m_1, ..., m_k$ such that $1 \leq m_j \leq n_j$, j=1,2,3,...,k and

$$m_A(\lambda) = (\lambda - \lambda_1)^{m_1}...(\lambda - \lambda_k)^{m_k}.$$

- If the minimial polynomial of L is

$$m(\lambda) = (p_1(\lambda))^{n_1}...(p_k(\lambda))^{n_k}$$

  Where $p_i(\lambda)$ are relatively prime, distinct monic irreducible polynomials
- Let λ be an eigenvalue of A $\epsilon M(K)$ and k be the multiplicity of λ.Then

$$1 \leq dim E_\lambda \leq K$$

2.1.3 Diagonalization and Triangularization

2. **Differential Equation**
3. **Probability**
4. **Signal System**
5. **FeedBack Control System**

X Programming

1. **C/C++ Program**
2. **Python Program**
   2.1 **Numpy**
   2.2 **Mathplotlib**
   2.3 **Scripy**
   2.4 **Casadi**
   2.5 **Acado**
3. **ROS2**

3.1 What is ROS?

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms." - according to the official ROS webpage.

3.2 Why you use ROS?

- ROS provides functionality for hardware abstraction, device drivers, communication between processes over multiple machines, tools for testing and visualization, and much more.
- All you have to do is to learn the concept of communication between all the nodes of the program, then, you'll be able to set up new parts of an application very easily.
- You need to compute a trajectory for your robot ? ROS has a package for that. You wish to use your joystick to control the robot ? ROS also has a package for that. You want to map out a room with a drone ? Guess what? ROS is the answer again.
- You can easily communicate between a Python node and a C++ node. It means a lot of reusability and possibilities of collaborative works. Many libraries also allow you to use other languages (because ROS has mainly targeted C++ and Python).
- Getting your robot to run for real can be complicated sometimes, therefore you certainly need simulation tools. ROS has many great tools, such as Rviz and Gazebo. For instance, Gazebo helps you add some physical constraints to the environment, so that once you run both simulation and physical robot, the outcome is pretty much identical.
- Control multiple robots
- When you build a robot, you don't necessarily wish to reinvent or recreate every part of it. You might want to focus on some development points, and integrate the rest from other manufacturers. The good news here is that you can find many robotics products, such as grippers or controller boards, that already have a ROS package. So, in addition to the physical tool, the software that goes with it is directly compatible with your ROS system.

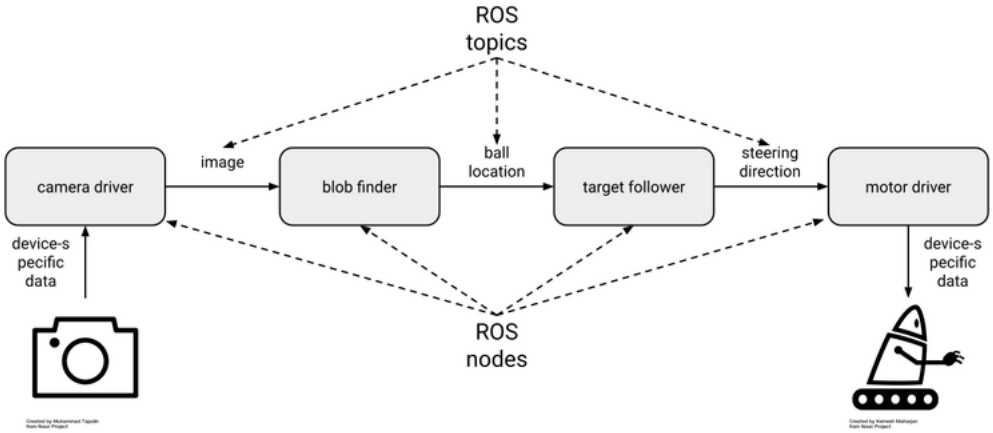
3.3 How to Setup ROS2?

Follow this link:`https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Debians.html`

3.4 ROS Concept and Design Pattern

3.4.1 The ROS Communication graph

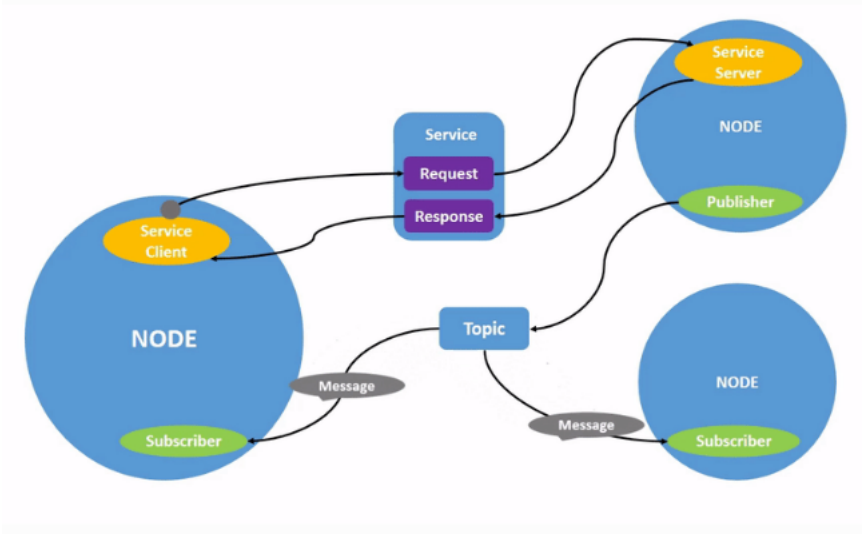
Imagine we are building a wheeled robot that chases a red ball. This robot needs a camera with which to see the ball, a vision system to process the camera images to figure out where the ball is, a control system to decide what direction to move, and some motors to move the wheels to allow it to move toward the ball. Using ROS we might construct the system like so:



This design separates the software into four ROS nodes: two device drivers and two algorithms. Those nodes communicate with each other as shown, via three ROS topics. We call this structure a ROS communication graph: the nodes are the graph vertices and the topics are the graph edges. You can tell a lot about a ROS system by examining its communication graph.

3.4.2 Node & Topics

- Each node in ROS should be responsible for a single, modular purpose, e.g. controlling the wheel motors or publishing the sensor data from a laser range-finder. Each node can send and receive data from other nodes via topics, services, actions, or parameters.
- Topics are a vital element of the ROS graph that act as a bus for nodes to exchange messages.



- Type of Topics which often used
  _from geometry_msgs.msg import Twist, Vector3
  _from std_msgs.msg import Float32MultiArray, UInt16MultiArray, Int8, Int16, Int32, UInt16, UInt32, String
  _from sensor_msg.msg import Imu, Joy, JoinState, Image, FluidPressure, LaserScan, Temperature,...
  _from nav_msgs.msg import Message(Odometry, Path GridCells, MapMetaData) Service(GetMap,GetPlan)

3.4.3 Publisher & Subscribe

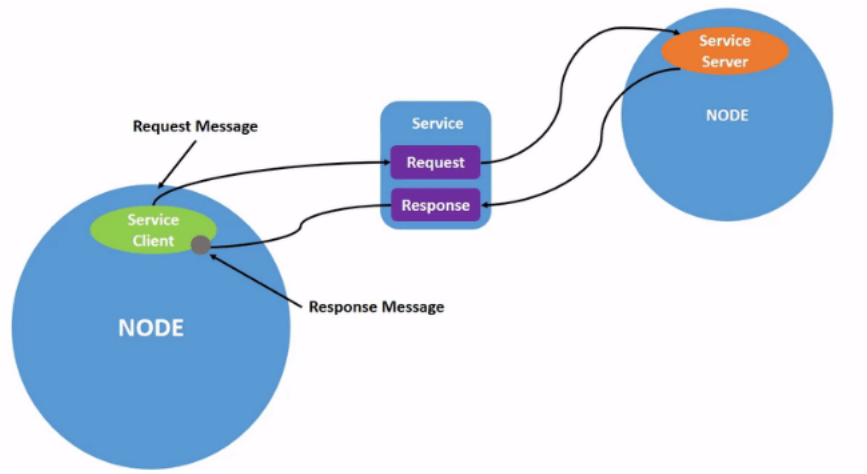
Given that it comes with additional complexity (nodes, topics, types, etc.), it is reasonable to ask why ROS follows the pub-sub pattern. After more than a decade

of building and deploying ROS-based robot systems, we can identify several key benefits:

- **Substitution:** If we decide to upgrade the robot's camera, we need only modify or replace the camera driver node. The rest of the system never knew the details of the camera anyway. Similarly, if we find a better blob finder node, then we can just swap it in for the old one and nothing else changes. **Reuse:** A well-designed blob finder node can be used today on this robot to chase the red ball, then reused tomorrow on a different robot to chase an orange cat, and so on. Each new use of a node should require only configuration (no code) changes.
- **Collaboration:** By cleanly separating concerns between nodes, we let our blob finder expert do their work independently of the target follower expert, with neither of them bothering the device driver expert. It is often the case that a robot application requires the combined expertise of many people, and it would be difficult to overstate the importance of ensuring that they can each contribute confidently and efficiently.
- **Introspection:** Because the nodes are explicitly communicating with each other via topics, we can listen in. So when the robot fails to chase the red ball, and we think that the problem is in the blob finder, we can use developer tools to visualize, log, and play back that node's inputs and outputs. The ability to introspect a running system in this way is instrumental to being able to debug it.
- **Fault tolerance:** Say that the target follower node crashes because of a bug. If it is running in its own process, then that crash will not bring down the rest of the system, and we can get things working again by just restarting the target follower. In general with ROS we have the choice to run nodes in separate processes, which allows for such fault tolerance, or run them together in a single process, which can provide higher performance (and of course we can mix and match the two approaches).
- **Language independence:** It can happen that our blob finder expert writes their computer vision code in C++, while our target follower expert is dedicated to Python. We can accommodate those preferences easily by just running those nodes in separate processes. In ROS, it is perfectly reasonable, and in fact quite common, to mix and match the use of languages in this way.

3.4.4 Service

Services are another method of communication for nodes in the ROS graph. Services are based on a call-and-response model versus the publisher-subscriber model of topics. While topics allow nodes to subscribe to data streams and get continual updates, services only provide data when they are specifically called by a client.



We might implement the new high-res snapshot capability like so:

- **Define a new service type:** Because services are less widely used than topics, there are relatively few "standard" service types predefined. In our case, the new service's request message might include the desired resolution of the snapshot. The request message could be a standard sensor_msgs/Image.
- **Implement the service:** In the camera driver, we would advertise the newly defined service so that when a request is received, the usual image-handling is interrupted temporarily to allow the device interaction necessary to grab one high-res snapshot, which is then packed into a reply message and sent back to the node that called the service.
- **Call the service:** In the target follower node, we might add a timer so that every 5 minutes, it calls the new service. The target follower would receive the high-res snapshot in response to each call, and could then, say, add it to a photo gallery on disk.

4. **MATLAB**
5. **STM32**