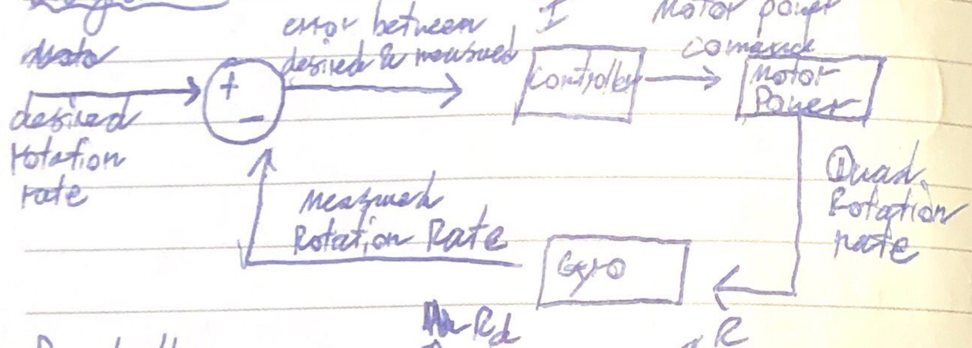
Proportional – Integral – Derivative

PID Control Loop Working

Need this logic

Diagram



P controller

$\text{Input}_{motor} = P \cdot (\text{Desired Rate} - \text{Rate})$

↳ 1000 – 2000 μs $\qquad \underbrace{-75°/s \Rightarrow 75°/s}$

$$\text{Error}(n) = R_d(n) - R(n)$$

↳ $\text{Input}_{motor}(n) = P \cdot (\text{Error}(n))\ [°/s]$

$T_s = 0.004\,s$ → 250 Hz

I controller

$$\text{Input}_{motor} = P \cdot \text{Error}(n) + I \int_0^{nT_s} \text{Error}(t)\,dt$$

• ~~Discretize~~

• Discretize the integral

past error $\qquad I \cdot I_{term}(k)$ current I average

↳ $I_{term}(k-1) + I \cdot \dfrac{(\text{Error}(k) + \text{Error}(k-1)) \cdot T_s}{2}$

↳ $In_{motor}(n) = P \cdot \text{Error}(n) + I_{term}(k-1) + I \cdot I_{term}(k)$

~~P shifts up~~ → shifts to desired

~~I Removes oscillations~~

~~D Diminishes overshoot~~

$$\frac{f(x+h) - f(x)}{h}$$

D controller

$Input_{motor} = P_{term}(k) + I_{term}(k) + D_{term}(k)$

$\hookrightarrow D \cdot \frac{d}{dt} Error(t) \rightarrow \frac{D \cdot (Error(n) - Error(n-1))}{T_s}$

PID Controller • $T_s = 250Hz$ or $0.004S$

~~$In_{m} = P(Desired_{Rate} - Rate)$ +~~ $\rightarrow Error(n)$

$In_{m} = P \cdot (R_d(n) - R(n))$
$+ I_{term}(n-1) + \frac{I \cdot (Error(n) + Error(n-1)) \cdot T_s}{2}$
$+ \frac{D \cdot (Error(n) - Error(n-1))}{T_s}$

In Arduino...

$In_R(n) = P \cdot Error$
$+ PrevI_{term} + I \cdot \frac{(Error + PrevError) \cdot T_s}{2}$
$+ D \cdot \frac{(Error - PrevError)}{T_s}$

• Need this for Pitch Roll & Yaw

:-(

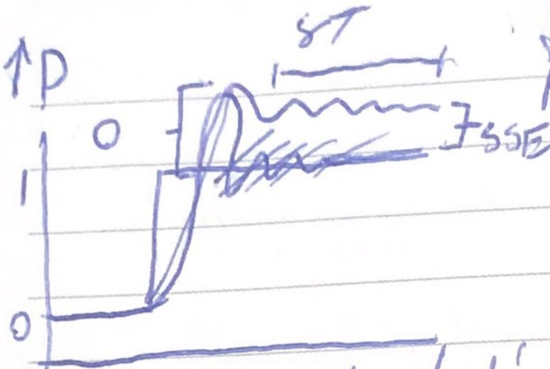
PID GRAPHS

O = Overshoot

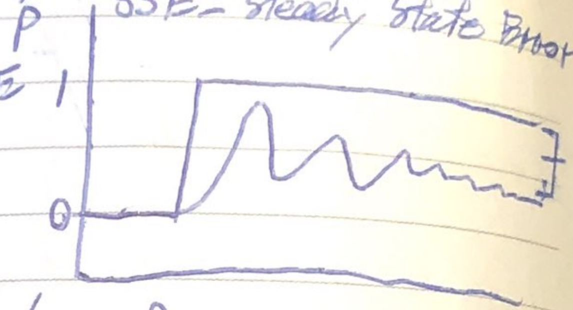
ST = steady ~~state~~

SSE = steady state Error

↑P

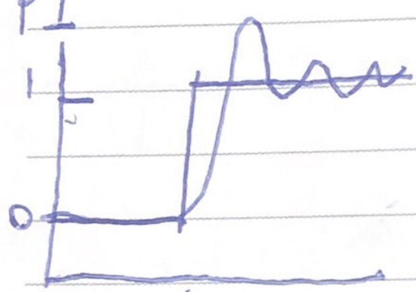


↓P



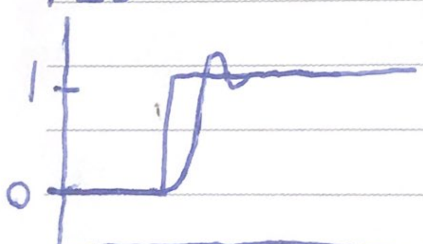
- P is overshoot, higher P means less setting time but more overshoot

PI



- I term removes steady state Error

PID



- D less settling time

Kalman Filter

- Combines previous noisy data & current noisy data to make a state estimate of a system

- For example, we have a car going down the road @ a relatively consistent speed repersented by:

$$\boxed{X_{t+1} = [A]\vec{X}_t + U, \quad U \sim N(0, P)}$$

 - Where X_t is the cars position
 - Where [A] is a numerical Matrix
 - Were U is the noise factor dependent on a known coveriance Matrix P

- But say we have a GPS or device that gives us the cars position as well repersented by:

$$\boxed{Y_t = [H]\vec{X}_t + V, \quad V \sim N(0, Q)}$$

 - Where X_t is the cars position
 - Where H is a numerical Matrix
 - Were V is the noise factor dependent on a Kown coveriance Matrix V

• Kalman is a two step Process

- Prediction step:

$X_t \longrightarrow A \longrightarrow \bar{X}_{t+1}$ • Idealized conversion

$\Sigma_t \longrightarrow A \longrightarrow \bar{\Sigma}_{t+1} = P$ • Noise as well

↳ $P + A\Sigma_t + A^T$

- **Update step**

• combine noise & X_t estimation with y_t observation

• In order to do this, we need to determine which measurement we prefer/trust more by comparing coveriance Matrix P & Q between the two represented by:

$$K_x = \frac{\Sigma_{xx}}{\Sigma_{xx} + Q_{xx}}$$

- Were K_x is the kalman value
- Σ_{xx} is the uncertainty matrix represented by Σ_{t+1} in this example
- Q_{xx} is y_t coveriance Matrix

So we combine these two equations & determine our surprise factor

$$\begin{matrix} \bar{x}_{t+1} \\ \bar{\Sigma}_{t+1} \end{matrix} \longrightarrow \text{(Update)} \xleftarrow{x_t} \longrightarrow x^{+}_{t+1} = x^{-}_{t+1} + K_x \Delta_x$$

- Where Δx is our surprise factor (Difference between two measurements

• But now we want to determine the cars speed from its position?

• Using the covariance matrix:

$$\Sigma = 0 \begin{bmatrix} \Sigma_{xx} & \Sigma_{x\dot{x}} \\ \Sigma_{x\dot{x}} & \Sigma_{\dot{x}\dot{x}} \end{bmatrix}$$ Uncertainty about state of system

- Σ_{xx} Uncertainty about position of car
- $\Sigma_{\dot{x}\dot{x}}$ Uncertainty about velocity of car
- $\Sigma_{x\dot{x}}$ Represent corelation between noise in measurement to the speed & position of the car

So to determine $\dot{x}$ all we have to do is change the kalman coefficient accordingly:

$$\dot{x}^+_{t+1} = \dot{x}^-_{t+1} + k_{\dot{x}} \Delta x$$

Where $k_{\dot{x}} = \dfrac{\Sigma_{x\dot{x}}}{\Sigma_{xx} + Q_{xx}}$

More generally:

$$x_{t+1} = Ax_t + U, \quad U \sim N(0, P)$$
$$y_t = Hx_t + V, \quad V \sim N(0, Q)$$

$$\left.\begin{matrix} x^-_{t+1} \\ \Sigma^-_{t+1} \end{matrix}\right\} \longrightarrow \text{update} \longrightarrow \begin{aligned} x^+_{t+1} &= x^-_{t+1} + k(y_{t+1} - Hx^-_{t+1}) \\ \Sigma^+_{t+1} &= (I - kH)\Sigma^-_{t+1} \end{aligned}$$

Measuring Angles with MPU-6050

- This is for more accurate measurement of ~~[illegible]~~ acceleration & height.
- This is because right now the flight controller responds to angles on a per second ratio, wanna change this to 10° pitch = 10° on joystick

-Math

$$\text{Angle Pitch} = \int_0^{kT_s} \underbrace{\text{Rate}_{pitch}}_{\text{degree/s}} \cdot dt$$

↳ $\text{AngPitch} = \text{degree} \ln(kT_s) - \text{degree} \ln(0)$

$\rightarrow 1$

- cant represent this so need to use logic

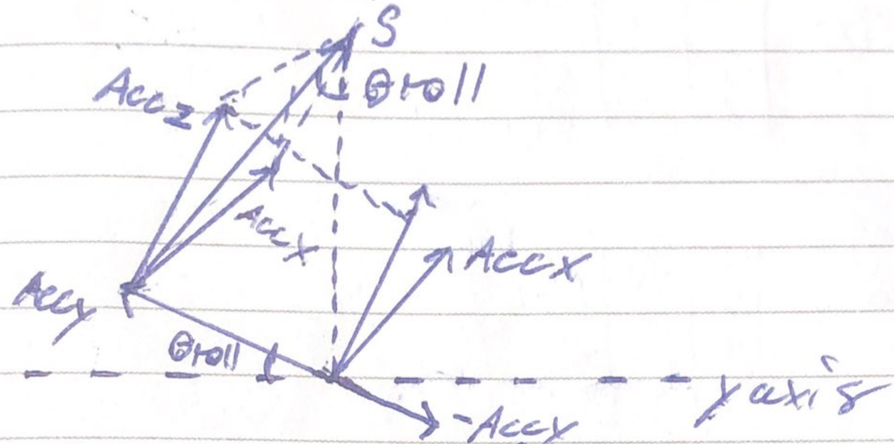
↳ $\text{Angle Pitch}(k) = \text{Angle}_{pitch}(k-1) + \text{Rate}_{pitch}(k) \cdot T_s$

↳ Previous + current

- However this will keep dragging measurement errors

- ~~With use accelero~~ Also doesn't allow us to account for Pitch & Roll changes with Yaw @ same time

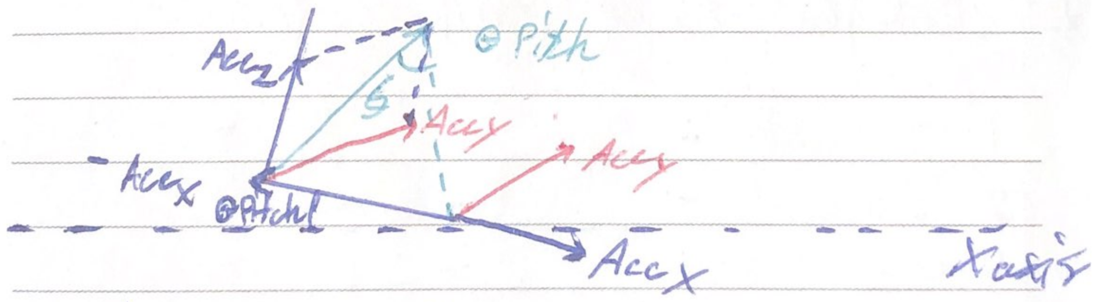
- will use accelerometer to counter this
- More Math - Roll around the x axis



- So $\tan(\theta_{roll}) = \frac{Acc_y}{S}$, $S = Acc_x^2 + Acc_z^2$

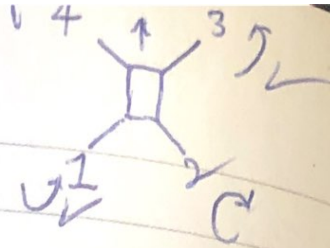
↳ $\tan(\theta_{roll}) = \frac{Acc_y}{\sqrt{Acc_x^2 + Acc_z^2}} \rightarrow \theta_{roll} = \tan^{-1}\left(\frac{Acc_y}{\sqrt{Acc_x^2 + Acc_z^2}}\right)$

- More Math - Pitch Around the Y Axis



- So $\tan(\theta_{pitch} = \frac{-Acc_x}{S}$

or $\theta_{Pitch} = \tan^{-1}\left(\frac{-Acc_x}{\sqrt{Acc_y^2 + Acc_z^2}}\right)$



Steering DRONE

Throttle

$$TU = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} 75\% \\ 75\% \\ 75\% \\ 75\% \end{bmatrix}$$

Right ROLL $= \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} 25\% \\ 75\% \\ 75\% \\ 25\% \end{bmatrix}$

Pitch = DOWN $\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} 75\% \\ 75\% \\ 25\% \\ 25\% \end{bmatrix}$

Yaw = CCW $\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} 25\% \\ 75\% \\ 25\% \\ 75\% \end{bmatrix}$

LINEAR COMBINATION

1 = Throttle - ____ + ____ - ____

2 = Throttle + ____ + ____ + ____

3 = Throttle + ____ - ____ - ____

4 = Throttle - ____ - ____ + ____

Updated Flight Controller

- current one is a degree per second response for pitch Roll & Yaw
  - i.e. if we hold pitch @ 45°
  it will continue to pitch 45°/s
  so @ t=1, 45°
  t=2 90°
  t= 135°...
- But we want it to be, hold stick @ 45° quadcopter stays @ 45°

- Updated PID Loop