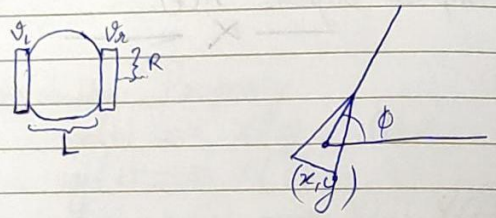
DIFFERENTIAL DRIVE ROBOT



$$\dot{x} = \frac{R}{2}(v_r + v_l)\cos\phi$$

$$\dot{y} = \frac{R}{2}(v_r + v_l)\sin\phi$$

$$\dot{\phi} = \frac{R}{L}(v_r - v_l)$$

* But it is not very natural to "think" in terms of wheel velocities.
* So go directly for translational & angular velocities.
* This model is called Unicycle Model.
* Inputs :- v velocity, ω angular velocity

* Dynamics.

$$\dot{x} = v\cos\phi$$
$$\dot{y} = v\sin\phi$$
$$\dot{\phi} = \omega$$

* In order to map both models together, equate $\dot{x}, \dot{y}, \dot{\phi}$ on both models.

* $v = \frac{R}{2}(v_r + v_l) \Rightarrow \frac{2v}{R} = v_r + v_l$

* $\omega = \frac{R}{L}(v_r - v_l) \Rightarrow \frac{\omega L}{R} = v_r - v_l$

* Solving both.

$$v_r = \frac{2v + \omega L}{2R}$$

$$v_l = \frac{2v - \omega L}{2R}$$

* Odometry is the means by which we can obtain this pose information. ~~and the~~

* The state of the robot is (x, y, ϕ)

* How to know how far ~~the~~ each wheel has moved.

- Assume each wheel has N "ticks" per revolution.
- For both wheels.

$$\Delta tick = tick' - tick$$

- $$Distance\ (D) = \frac{2\pi R\, \Delta tick}{N}$$

---

* While dealing with angles having high multiples of π cause errors
* A way to convert the value & ensure that $e \in [-\pi, \pi]$
* $e' = atan2(\sin(e), \cos(e)) \in [-\pi, \pi]$

  This is equivalent to $e' = \tan^{-1}\left[\frac{\sin(e)}{\cos(e)}\right]$
* To drive a robot to a goal location

$$\begin{cases} \dot{x} = v_o \cos\phi \\ \dot{y} = v_o \sin\phi \\ \dot{\phi} = \omega \end{cases} \qquad \begin{aligned} e &= \phi_d - \phi \\ \omega &= PID(e) \end{aligned}$$

* For inputs $v = 0$, $\omega = const$, find the corresponding wheel velocities v_r, v_l

$$v_r = \frac{2v + \omega L}{2R} \quad ; \quad v_l = \frac{2v - \omega L}{2R}$$

For the conditions

$$v_r = \frac{cL}{2R} \quad ; \quad v_l = \frac{-cL}{2R} \qquad \{\text{Where } c = \omega \text{ and constant}\}$$

* Hence in above case the ~~syst~~ rover will only rotate in place.

* Wheel encoders.
- Total N ticks per revolution
- $D_l = \frac{2\pi R\, \Delta tick_l}{N}$
- $D_r = \frac{2\pi R\, \Delta tick_r}{N}$

  } dt
- $D_c = \frac{D_r + D_l}{2}$
- $x' = x + D_c \cos\phi$
- $y' = y + D_c \sin\phi$
- $\phi' = \phi + \frac{D_r - D_l}{L}$

* For a small period dt
  - $Dx = Rv_x\, dt$
  - $D_z = Rv_z\, dt$
  - $x(t+dt) = x(t) + \dot{x}\, dt$

—x—

* Let's recall our car model.

$$\dot{v} = \frac{c}{m} u - \gamma v$$

* If we want to measure velocity.

$$\dot{x} = Ax + Bu \qquad A = -\gamma,\ B = \frac{c}{m}$$
$$y = Cx$$

* If we care about the position we have the same general equation with different matrices:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -\gamma \end{bmatrix},\quad B = \begin{bmatrix} 0 \\ c/m \end{bmatrix},\quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

* The State Transition Matrix

$$e^{A(t-t_0)} = \phi(t, t_0)$$

* $\dot{x} = Ax$

$$\Rightarrow x(t) = \phi(t,\tau)\, x(\tau) = e^{A(t-t_0)}\, x(\tau)$$

* $\dfrac{d}{dt}\phi(t,t_0) = A\,\phi(t,t_0)$

$$\phi(t,t) = I$$

* But what if we have the controlled system

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

* Sdn^n of this differential eq^n

$$x(t) = \phi(t,t_0)\, x(t_0) + \int_{t_0}^{t} \phi(t,\tau)\, Bu(\tau)\, d\tau$$

—x—

* Stability

$$\dot{x} = Ax \Rightarrow x(t) = e^{At} x(0)$$

  - Asymptotically Stable (if & only if)
    $\text{Real}(\lambda) < 0,\ \forall\ \lambda \in \text{eigenvalue}(A)$
  - Unstable
    $\exists\ \lambda \in \text{eig}(A) : \text{Re}(\lambda) > 0$
  - Critically Stable
    $\text{Re}(\lambda) \leq 0,\ \forall\ \lambda \in \text{eig}(A)$

- Critical Stable (if): One eigenvalue = 0 & rest have -ve real part OR 2 pure imaginary eigenvalues & rest have -ve real part.

- Fact: If one eigenvalue is 0 & all others have negative real part, then the state will end up in the so-called null-space of A

$$\text{null}(A) = \{x : Ax = 0\}$$

- For this particular A, the null-space is

$$\text{null}(A) = \{x : x = \begin{bmatrix} \alpha \\ \alpha \end{bmatrix}, \alpha \in \mathbb{R}\}$$

- $x_1 \to \alpha$, $x_2 \to \alpha \iff (x_1 - x_2) \to 0$
- Rendezvous is achieved.

- If more than 2 agents, they should aim towards the centroid of neighbours.

$$\dot{x}_i = \sum_{j=N_i} (x_j - x_i)$$

- $x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$ $\quad \dot{x} = -Lx$

— x —

- Closing the loop

$$\dot{x} = Ax + Bu$$
$$u = -Kx$$
$$\dot{x} = Ax + Bu = Ax - BKx$$
$$= \underbrace{(A - BK)x}_{\text{Closed loop dynamics}}$$

- Pick, if possible, K such that the closed loop system is stabilised ie.

$$Re(eig(A - BK)) < 0$$

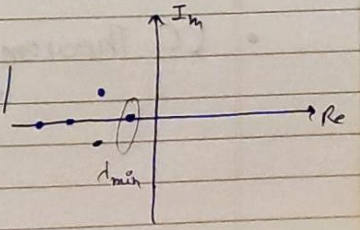
— x —

- Rate of Convergence

$$\dot{x} = (A - BK)x, \quad Re(\lambda) < 0, \forall \lambda \in eig(A - BK)$$

- The "smallest" eigenvalue dominates the convergence rate

$$\lambda_{min} = \arg\min_{\lambda} |Re(\lambda)|$$



- But bigger the eigen values, bigger the control gains / signals.

- There is a ~~the~~ problem though with bigger values. If you make them really large, you get really large control signals, ~~such~~ & any physical actuator is going to saturate.

* Controllability – Theorem 1

- Def^n: The system is completely controllable (CC) if it is possible to go from any initial state to any final state.

$$T = [B \quad AB \quad \ldots \quad A^{n-1}B]$$ ← Controllability Matrix.

- CC Theorem 1: The system is CC if and only if rank $(T) = n$

- rank (M) = # linearly independent raws or column in M.

- ~~Controllability~~

- CC Theorem 2: Pole-Placement to arbitary eigenvalues is possible if & only if the system is CC!

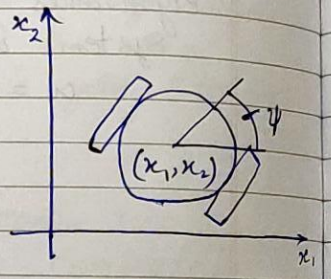
$$u = -Kx \;, \quad \dot{x} = (A - BK)x$$

SEGWAY

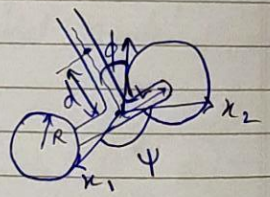
* Basically is a combination of a bunch of systems
  → Unicycle + Inverted Pendulum + ...

* The Base

$$\dot{x}_1 = v\cos\psi$$
$$\dot{x}_2 = v\sin\psi$$
$$\dot{\psi} = \omega$$



* The Pendulum —
  ϕ, $\dot{\phi}$
  Input : τ_L, τ_R
  Extra state : v, ω



State of System — $x = [x_1 \;\; x_2 \;\; v \;\; \psi \;\; \dot{\psi} \;\; \phi \;\; \dot{\phi}]^T$

Input — $u = [\tau_R \;\; \tau_R]^T$

$$\dot{x} = f(x,u)$$
$$= [v\cos\psi \;\; v\sin\psi \;\; \dot{v} \;\; \dot{\psi} \;\; \ddot{\psi} \;\; \dot{\phi} \;\; \ddot{\phi}]^T$$

* On ~~chet~~ checking the rank of AB, it is found out to be not equal to order so not CC.
* However if we consider a smaller system
  $x = [v \;\; \omega \;\; \phi \;\; \dot{\phi}]^T$, $u = [\tau_L \;\; \tau_R]^T$

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

* We get a CC system
* Also we don't want $(v, \omega) = (0,0)$
* Let

$$\tilde{x} = x - \underbrace{[v_d \;\; \omega_d \;\; 0 \;\; 0]^T}_{\delta} = x - \delta$$

* The dynamics become

$$\dot{\tilde{x}} = \dot{x} - \cancel{\dot{\delta}}^{\,0} = \dot{x} = Ax + Bu = A(x-\delta) + Bu + A\delta$$

* But Luckily $A\delta = 0$
$\Rightarrow \dot{\hat{x}} = A\hat{x} + Bu$

* Therefore the curvature of the path
$k = \frac{\omega}{U}$

* Predictor Corrector.
$\dot{x} = Ax$
$y = Cx$
$\Rightarrow$ First Idea: Make a copy of the system.
$\dot{\hat{x}} = A\hat{x}$ predictor.

Second idea: Add a notion of how wrong your estimate is to the model

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

$\Rightarrow$ We want to stabilise estimation error.
$e = x - \hat{x}$

$\dot{e} = \dot{x} - \dot{\hat{x}} = Ax - A\hat{x} - L(y - C\hat{x})$

$\dot{e} = A(x - \hat{x}) - LC(x - \hat{x})$ $(\because y = Cx)$

$\dot{e} = (A - LC)e$

$\Rightarrow$ So pick L such that the eigenvalues to $[A - LC]$ (matrix) have -ve real part
stability criterion.

$\Rightarrow$ • We want
$Re(eig(A - LC)) < 0$

• Do this by Pole-Placement.

Eg. $\dot{x} = \begin{bmatrix} -1 & 2 \\ 0 & -2 \end{bmatrix} x$, $y = \begin{bmatrix} 1 & -1 \end{bmatrix} x$

$\dot{e} = (A - LC)e = \left(\begin{bmatrix} -1 & 2 \\ 0 & -2 \end{bmatrix} - \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} \right) e$

$= \begin{bmatrix} -1 - L_1 & 2 + L_1 \\ -L_2 & -2 + L_2 \end{bmatrix} e$

- Next we find the characteristic eq^n

$$\chi_{A-Lc}(\lambda) = \begin{vmatrix} \lambda+1+L_1 & +2-L_1 \\ L_2 & \lambda+2-L_2 \end{vmatrix}$$

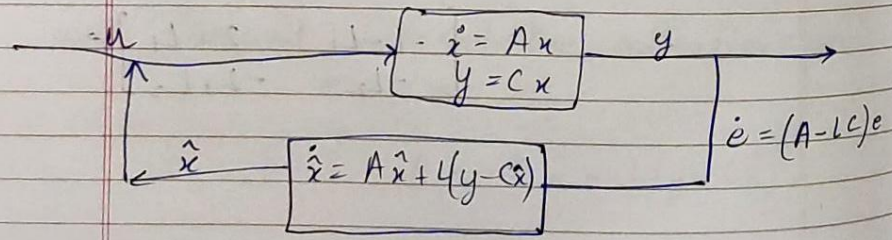
$$= \lambda^2 + \lambda(-3+L_1-L_2) + 2+2L_1+L_2$$

- Lets pick, say $\lambda = -1$
then
$\varphi(\lambda) = (\lambda+1)^2 = \lambda^2+2\lambda+1$

- Next line up the coefficients.
$L_1 = -2/3, \quad L_2 = 1/3$

$$\therefore \dot{\hat{x}} = A\hat{x} + \frac{1}{3}\begin{bmatrix} -2 \\ 1 \end{bmatrix}(y - C\hat{x})$$

BLOCK DIAGRAM



Observervability

- Given a discrete time system without inputs.

$$x_{k+1} = Ax_k$$
$$y_k = Cx_k$$

- $y_{n-1} = CA^{n-1}$

- $$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}}_{\Omega \rightarrow \text{Observability Matrix}} x_0$$

- The initial condition can be recovered from the outputs when the so-called observability matrix has full rank

- Observability Theorem 1
$\dot{x} = Ax, \quad x \in R^n$
$y = Cx$
$\Rightarrow$ The system is completely observable (CO) if it is possible to recover the

initial state from the output.

$$\Omega = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

⇒ The system is Completely observable if rank $(\Omega) = n$

- Observability Theorem 2

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

CO Theorem 2: Pole-placement to arbitrary eigenvalues is possible if & only if the system is CO

GAME PLAN

$\dot{x} = Ax + Bu$ (Assume CC & CO)

$y = Cx$

- Step 1: Design the state feedback controller as if we had x (which we don't)

$u = -kx$

⇒ $\dot{x} = (A - Bk)x$ ← what we design for

$u = -k\hat{x}$ ← what we actually have

- Step 2: Estimate x using an observer (that now also contains u)

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

⇒ $\dot{e} = (A - LC)e$ $\quad e = x - \hat{x}$

- Analysis

$\dot{x} = Ax - Bk\hat{x} = Ax - BK(x - e)$

$= (A - Bk)x + Bke$

$\dot{e} = (A - LC)e$

- Together.

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A-BK & BK \\ 0 & A-LC \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix}$$

- Separation Principle.

→ The above is an (upper) triangular block-matrix. Its eigenvalues are given by the eigenvalues of the diagonal blocks!

$$\chi_M(\lambda) = \chi_{A-BK}(\lambda)\, \chi_{A-LC}(\lambda)$$

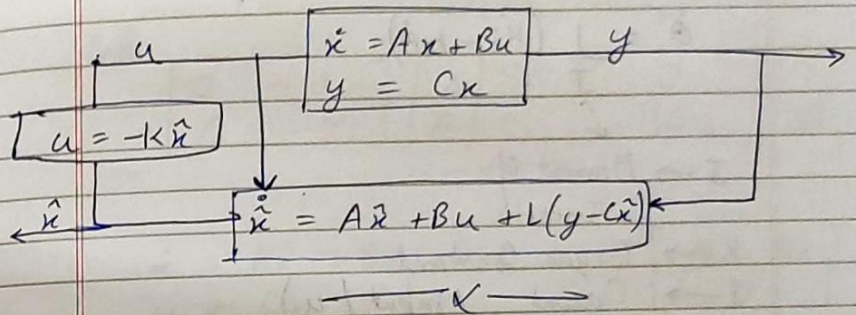
- We used pole placement to make sure that

$$Re(eig(A-BK)) < 0, \; Re(eig(A-LC)) < 0$$

- Thus by Separation Principle we can design controllers as if we have x.

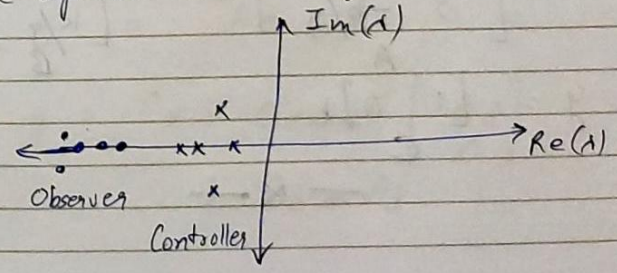
- Controllers & Observers are independent

- The final Block Diagram.



Eigenvalue Selection

- Typically, the observer should be faster than the controller, since the controller won't do anything useful until the state estimate is close.
- Note: Large eigenvalues give large observer gains: — Not a problem — The observer is a software construct!



* For the Arm motion of Humanoid

$$\ddot{\theta} = \frac{1}{J}(Ki - b\dot{\theta})$$

$J \rightarrow$ Moment of Inertia
$K \rightarrow$ Torque Constant
$i \rightarrow$ Current (input / u)
$b \rightarrow$ friction coefficient

$$\dot{x}_1 = \theta, \quad x_2 = \dot{\theta}$$
$$y = \theta.$$

* Linear Time-Invariance

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -b/J \end{bmatrix}_{A} x + \begin{bmatrix} 0 \\ K/J \end{bmatrix}_{B} u$$

$$y = \underbrace{[1 \quad 0]}_{C} x$$

— x —

Soln: How this come

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \frac{1}{J}(Ki - b\dot{\theta}) \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 0 & -b/J \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ K/J \end{bmatrix} i$$

— x —

$$\dot{x} = Ax + Bu \qquad y = Cx$$

$$e = \begin{bmatrix} \theta - \theta_d \\ \dot{\theta} \end{bmatrix}$$

$$\dot{e} = \begin{bmatrix} \dot{\theta} - \dot{\theta}_d \to 0 \\ \ddot{\theta} \end{bmatrix}$$

$$\Rightarrow \dot{e} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = x$$

$$\therefore \dot{e} = Ax + Bu$$
$$= Ae + A\begin{bmatrix} \theta_d \\ 0 \to 0 \end{bmatrix} + Bu$$
$$= Ae + Bu$$

$$y = Cx = Ce + C\begin{bmatrix} \theta_d \\ 0 \end{bmatrix}$$
$$= Ce + \theta_d$$

$$u = -K\hat{e}$$
$$\dot{\hat{e}} = A\hat{e} + Bu + L(y - C\hat{e} - \theta_d)$$

* Beyond Pole placement
  - $u = -Kx$ $\quad \dot{x} = A\vec{x} + Bu + L(y - C\vec{x})$
  - LQ Optimal Control: $\min \int_0^\infty (x^T Q x + u^T R u)\,dt$

$K = R^{-1} B^T P \qquad Q \geq 0,\ R > 0$

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

- L → The Kalman Filter.

$$\min \lim_{t \to \infty} \| E(x(t) - \hat{x}(t)) \|^2$$

White Noise

$$\dot{x} = Ax + Bu + D_1 v_1$$
$$y = Cx + D_0 v_0$$

Sol^n of the eq^n

$$L = PC^T (D_0 D_0^T)^{-1}$$
$$0 = A^T P + PA + D_1 D_1^T - PC^T (D_0 D_0^T)^{-1} CP$$

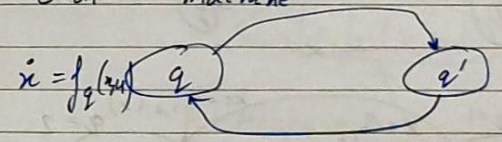
HYBRID AUTOMATA

* Modes, Transitions, Guards, & Resets.
  - Let the continuous state be x.
  - Another discrete state q, which will tell which different dynamic, continuous mode I am in.
  - Dynamics:

$$\dot{x} = f_q(x, u)$$

this show where I actually am.

  - The transitions b/w different ~~state machines~~ discrete modes can be encoded in a state machine



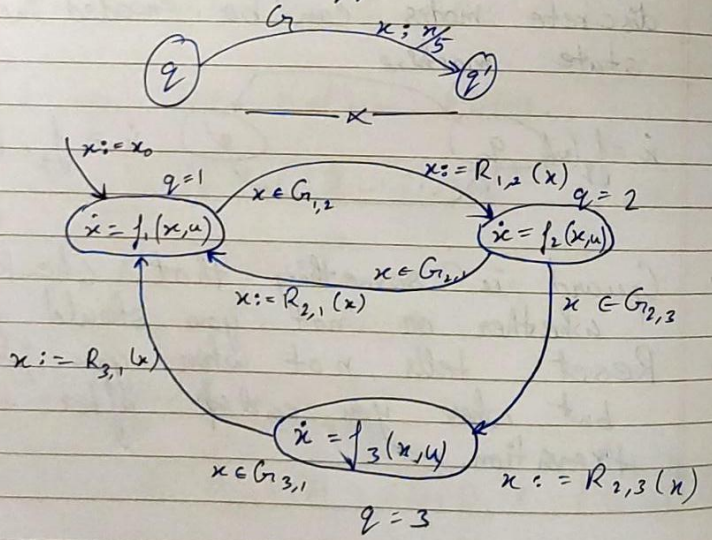
  - Guard is something that check whether or not you should jump
  - Reset tells not when you jump but where you end up after the transition.

HYBRID AUTOMATA

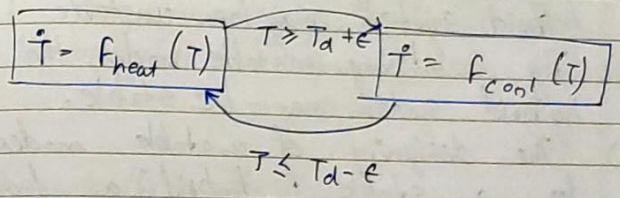
- The condition under which a transition occurs are called guard condition, ie, a transition occurs from q to q' if $x \in G_{q,q'}$

- As a final component, we would like to allow for abrupt changes in the continuous state as the transition occur, which we call reset

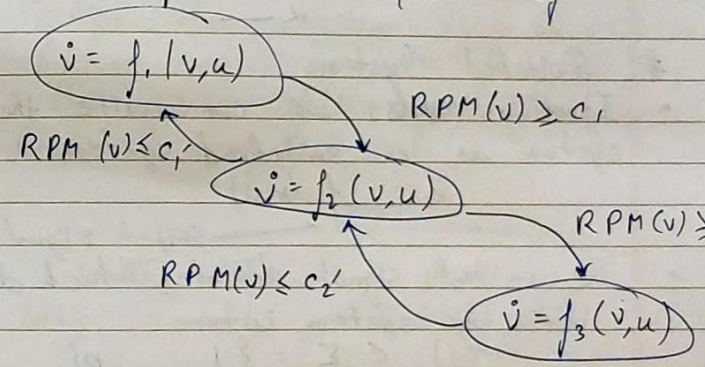
$x := R_{q,q'}(x)$



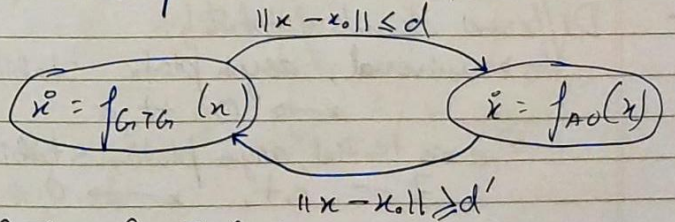
HA Example 1 – Thermostat



HA Example 2 – Gear Shift



HA Example 3 – Behaviours.



GTG → Go To Goal
AO → Avoid Obstacles

NOTE: 1) By combining stable modes, the resulting hybrid system may be unstable

2) By combining unstable modes, the resulting hybrid system may be stable.

3) The designing of unstable modes & hoping for a stable hybrid is a fool's choice & thus we will only design stable system

— x —

Switched Systems.

- Ignoring resets, we can write the hybrid system as a switched system:

$$\dot{x} = f_{\sigma}(x,u)$$

(σ ← switch signal)

- The switch signal dictates which discrete mode the system is in.

$$\sigma(t) \in \Sigma = \{1, \ldots, p\}$$

- Different kinds of stability

→ universal, asymptotic stability

$$x \to 0, \ \forall \sigma$$

→ existential, asymptotic stability:

$$\exists \sigma \text{ s.t. } x \to 0.$$

- If the switch signal is generated by an underlying hybrid automaton:

→ hybrid, asymptotic stability:

$$x \to 0$$

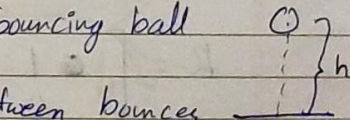
Practically Speaking.

- Design stabilizing controllers for the subsystems.
- Design the switching logic in HA (find common Lyapunov function)
- Be aware of the potential dangers here & test, test, test!

— x —

The Bouncing Ball.

* Let's model a bouncing ball



* Eq^{n} of motion in-between bounces

$$\ddot{h} = -g \Rightarrow \dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ -g \end{bmatrix}$$

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

$x_1 = h$

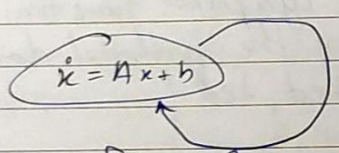
$x_2 = \dot{h}$

* Bounces.

$\dot{h} := -\gamma \dot{h} \Rightarrow x := \begin{bmatrix} 1 & 0 \\ 0 & -\gamma \end{bmatrix} x$

This is our reset condition.

* Lets see The Ball HA



$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ -g \end{bmatrix}$

$R = \begin{bmatrix} 1 & 0 \\ 0 & -\gamma \end{bmatrix}$

* Guards & Conditions

$h \leq 0 \Rightarrow \begin{bmatrix} 1 & 0 \end{bmatrix} x \leq 0$

Clearly we bounce when we hit the ground

* Guard Condition

$h \leq 0 \;\&\; \dot{h} \leq 0 \Rightarrow x \leq 0$

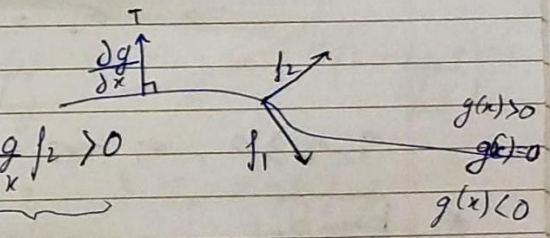
This is because ball will bounce only at ground

→ this $+0$ make sure we are facing downwards

* But this model is not good.

* Zeno phenomenon → Type 1 → ∞ switches in 0 time
Type 2 → ∞ switches in finite but not 0 time.

Sliding Control



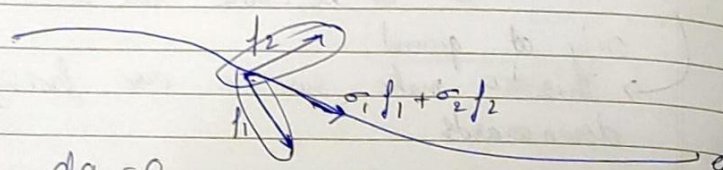
* Sliding occurs if

$\frac{\partial g}{\partial x} f_1 < 0 \;\&\; \frac{\partial g}{\partial x} f_2 > 0$

Means: point in different directions; point in same directions

$\frac{\partial g}{\partial x} f = \begin{bmatrix} \frac{\partial g}{\partial x_1} & \cdots & \frac{\partial g}{\partial x_n} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$

* derivative of g in direction f = $L_f g$ = Lie derivative



$$\frac{dg}{dt} = 0$$

$$\frac{dg}{dt} = \frac{\partial g}{\partial x}\dot{x} = \frac{\partial g}{\partial x}(\sigma_1 f_1 + \sigma_2 f_2)$$

$$= \sigma_1 L_{f_1} g + \sigma_2 L_{f_2} g = 0$$

$$\Rightarrow \sigma_2 = -\sigma_1 \frac{L_{f_1} g}{L_{f_2} g}$$

$\sigma_1, \sigma_2 \geq 0$, $\sigma_1 + \sigma_2 = 1$

* Example

$$\dot{x} = \begin{cases} -1 & x \geq 0 \\ +1 & x < 0 \end{cases} \qquad g(x) = x = 0$$

$$L_{f_1} g = \frac{\partial g}{\partial x} f_1 = 1(-1) = -1 < 0$$

$$L_{f_2} g = \frac{\partial g}{\partial x} f_2 = 1 \cdot 1 = 1 > 0$$

$\therefore$ Type 1 zeno.

$$\sigma_2 = -\sigma_1 \frac{L_{f_1} g}{L_{f_2} g} = -\sigma_1 \frac{-1}{1} = \sigma_1$$

$$\Rightarrow \sigma_2 = \sigma_1$$

Also $\sigma_1 + \sigma_2 = 1 \Rightarrow \sigma_1 = \sigma_2 = 0.5$

The Induced mode.

$$\dot{x}_i = 0.5 f_1 + 0.5 f_2 = 0$$

($f_1 \to -1$, $f_2 \to 1$)

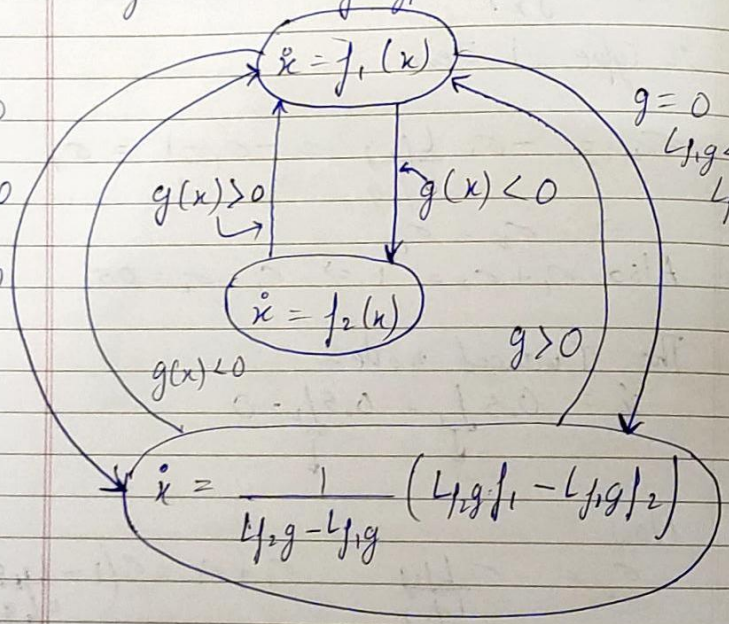
Now

$$\sigma_2 = -\sigma_1 \frac{L_{f_1} g}{L_{f_2} g} \qquad \sigma_1 + \sigma_2 = \sigma_1 \left(1 - \frac{L_{f_1} g}{L_{f_2} g}\right) = 1$$

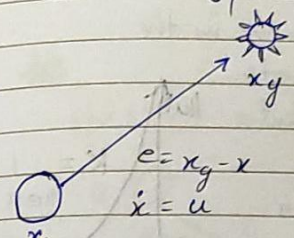
$$\Rightarrow \sigma_1 = \frac{L_{f_2} g}{L_{f_2} g - L_{f_1} g} \quad \& \quad \sigma_2 = \frac{-L_{f_1} g}{L_{f_2} g - L_{f_1} g}$$

$$\& \quad \boxed{\dot{x} = \frac{1}{L_{f_2} g - L_{f_1} g}\left(L_{f_2} g f_1 - L_{f_1} g f_2\right)}$$

* Regularization of Type 1 Zeno HA



Go-To-Goal



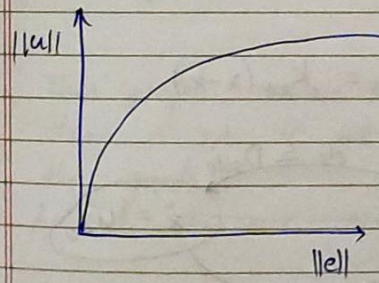
$u = Ke \Rightarrow \dot{e} = 0 - \dot{x} = -Ke$

Asymptotically Stable ?

$K > 0$ or $K \succ 0$ ($eig[K] > 0$)

$e \longrightarrow 0$

* A linear controller means the robot goes faster the further away the goal is.
* Solution, in practice, make the gain K a function of e



$\dot{e} = -K(\|e\|)e$

$K = \frac{v_0(1 - e^{-\alpha\|e\|^2})}{\|e\|}$

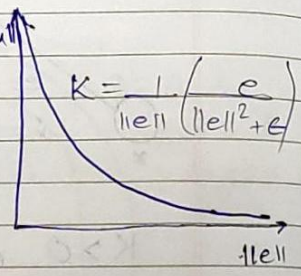
* Obstacle Avoidance.

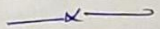
⇒ We don't want to move in the complete opposite direction.

⇒ We want k dependent on e!

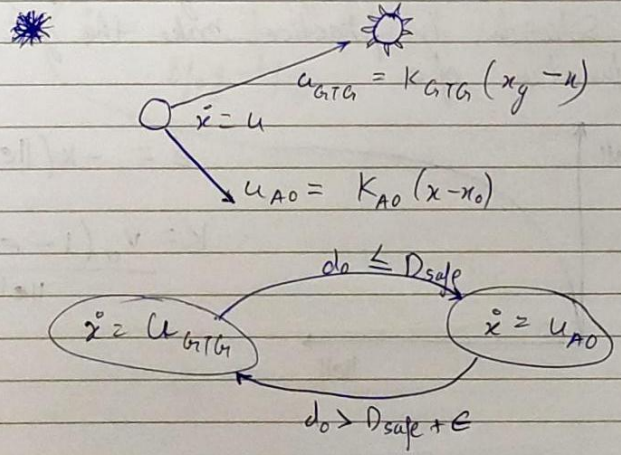
⇒ We will switch between behaviours.

⇒ ϵ is a small value so that k doesn't go off to ∞ at $e \to 0$

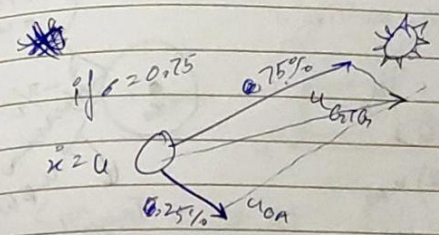




A Switch HA



Blending

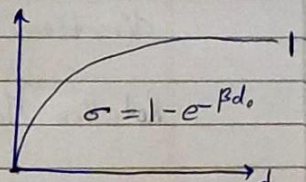


* Blending is like if $\sigma = 0.75$ then

$$\dot{x} = \sigma(d_o)\, u_{GTG} + (1 - \sigma(d_o))\, u_{OOA}$$

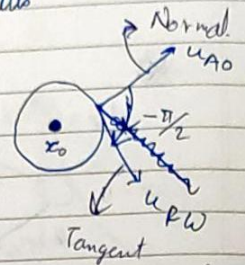
Example

* Of Blending question



* This graph means that as d_o is large then $\sigma = 1$, ie we are only going towards the goal.
* As d_o decreases we get a lower value of σ.

How To Follow Walls



Rotation matrix

$$u_{FW} = \alpha \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} u_{AO}$$

$$= \alpha R(-\pi/2)\, u_{AO}$$

$u_{FW} \rightarrow u^{c}_{FW}$ → clockwise $(-\pi/2)$

$u_{FW} \rightarrow u^{cc}_{FW}$ → counter-clockwise $(\pi/2)$

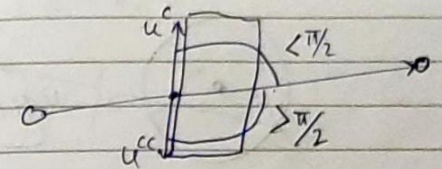
- We can clearly move in two different directions along a wall!

Rotation Matrix : $R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$

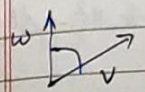
$$u^{c}_{FW} = \alpha R(-\pi/2)\, u_{AO} = \alpha \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} u_{AO}$$

$$u^{cc}_{FW} = \alpha R(\pi/2)\, u_{AO} = \alpha \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} u_{AO}$$

Clockwise or Counter-Clockwise.



$$\langle v, w \rangle = v^{T} w = \|v\| \|w\| \cos\left(\angle(v,w)\right)$$



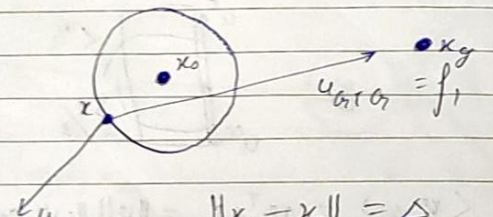
if the ~~angle is acute~~ dot product is
- -ve then less than $\pi/2$
- +ve more than $\pi/2$

Also we want the dot product to be +ve so that we are moving down.

$\therefore \langle u_{GTG}, u^{c}_{FW} \rangle > 0 \Rightarrow u^{c}_{FW}$

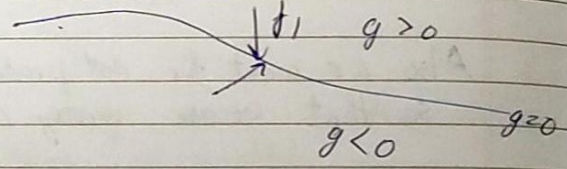
$\langle u_{GTG}, u^{cc}_{FW} \rangle > 0 \Rightarrow u^{cc}_{FW}$

The Setup



$$\|x - x_0\| = \Delta$$

$$g(x) = \frac{1}{2}\left(\|x - x_0\|^2 - \Delta^2\right) = 0$$



$$f_1 = C_{G_1+G_1}(x_g - x)$$

$$f_2 = C_{A0}(x - x_0)$$

The induced mode:

$$\dot{x} = \frac{1}{L_{f_2} g - L_{f_1} g}\left(L_{f_2} g f_1 - L_{f_1} g f_2\right)$$

$$\frac{\partial g}{\partial x} = (x - x_0)^T$$

$$L_{f_2} g = \frac{\partial g}{\partial x} f_2 = (x - x_0)^T C_{A0}(x - x_0)$$

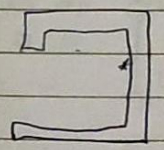
$$= C_{A0}\|x - x_0\|^2$$

$$L_{f_1} g = \frac{\partial g}{\partial x} f_1 = C_{G_1+G_1}(x - x_0)^T (x_g - x)$$

The Induced Mode:

$$\dot{x} = \frac{1}{L_{f_2} g - L_{f_1} g}\left(L_{f_2} g f_1 - L_{f_1} g f_2\right)$$

When to stop



τ = time of last switch

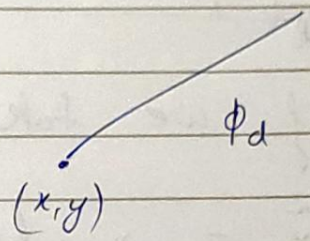
progress: $\|x - x_g\| < \|x(\tau) - x_g\|$

clear shot: $\langle u_{A0}, u_{G_1+G_1} \rangle \geq 0$

Dealing with angles.

- How to drive the robot in a specific direction?

$$\dot{x} = v\cos\phi$$
$$\dot{y} = v\sin\phi$$
$$\dot{\phi} = \omega$$



$$e = a^y$$

$$e = \phi_d - \phi, \qquad \omega = PID(e)$$

$$PID(e) = K_p e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \dot{e}(t)$$

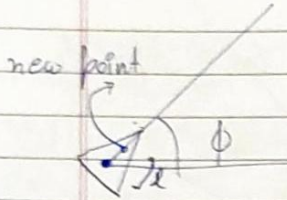
$$e = a\tan 2(\sin(\phi_2 - \phi), \cos(\phi_2 - \phi)),$$

- Adding in a speed component.
- Let the output from the planner be

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad (\dot{x} = u)$$

$$\phi_d = \operatorname{atan}\left(\frac{u_2}{u_1}\right)$$

$$\sqrt{\dot{x}^2 + \dot{y}^2} = \sqrt{v^2\cos^2\phi + v^2\sin^2\phi}$$
$$= v$$



$\dot{x} = v\cos\phi$

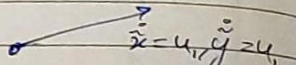
$\dot{y} = v\sin\phi$

$\dot{\phi} = \omega$

* If we take another point on it then

$\tilde{x} = x + l\cos\phi$

$\tilde{y} = y + l\sin\phi$



$\dot{\tilde{x}} = \dot{x} - l\dot{\phi}\sin\phi = v\cos\phi - l\omega\sin\phi = u_1$

$\dot{\tilde{y}} = \dot{y} + l\dot{\phi}\cos\phi = v\sin\phi + l\omega\cos\phi = u_2$

$$\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} v \\ l\omega \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$= R(\phi) \begin{bmatrix} 1 & 0 \\ 0 & l \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

We know that

$$R^{-1}(\phi) = R(-\phi)$$

$$\begin{bmatrix} 1 & 0 \\ 0 & l \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{l} \end{bmatrix}$$

$$\therefore \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{l} \end{bmatrix} R(-\phi) \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$