# Lecture 3 – ~~Dynamic Programming~~ (Intro to Optimal Control)

**Till now**

What do we want the acc$^n$ to be at every $q$.?
That's short-sighted, means we are going to do instantaneously change our dynamics.

→ If we want to do long-term thinking then we have to ~~do something that~~ write down the problem in a way that think about a long-term evolution of the dynamics.

RL is a learning based version of optimal control

So, instead of saying show me ~~good~~ a vector field and we will tell if it's good or not, we have to say now instead — show me an entire trajectory and we will tell if it's good or not.

**Longer term view of the world**

Given a trajectory $x(0), u(0)$ shorthand for $x(t)\ \forall t \in [t_0 \dots t_f]$, $u(t)\ \forall t \in [t_0, \dots, t_f]$ ($t_f$ = final)

Assign a scalar cost/reward.
Goal of control then is to minimize the long-term cost.

RL → max$^m$ reward
Control theory → min$^m$ cost

→ Can also add constraints:

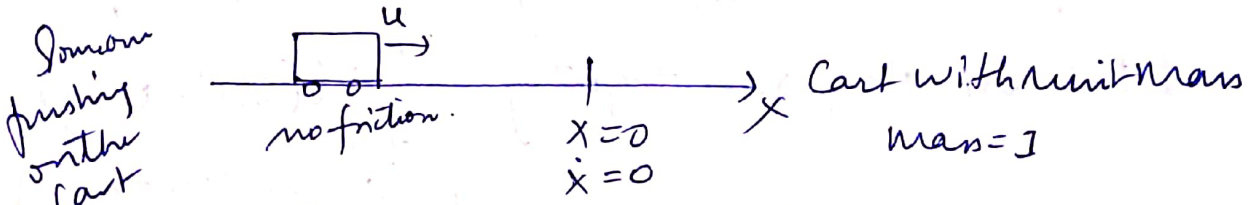
$$\forall t \quad |u(t)| \leq u_{max} \quad \text{(torque limit)}, \quad x(t_f) = x_{goal}$$

($t_f$ = finite (at final time, robot gets to the goal))

Among trajectories that get to the goal, we want to pick the best — the policy that works best.

## Minimum time problem for Double Integrator

$$\ddot{q} = u \quad , \quad |u| \leq 1$$




Goal ⇒ from any initial condition we want to write a controller that gets to the origin as fast as possible in the minimum time.

Max acch.

(1) Case – Deceleration part

Slamming on the brakes on reaching the goal.

$$\ddot{q} = u$$
$$u = -1$$
$$\dot{q} = \dot{q}(0) - t$$
$$q(t) = q(0) + t\dot{q}(0) - \frac{1}{2}t^2$$

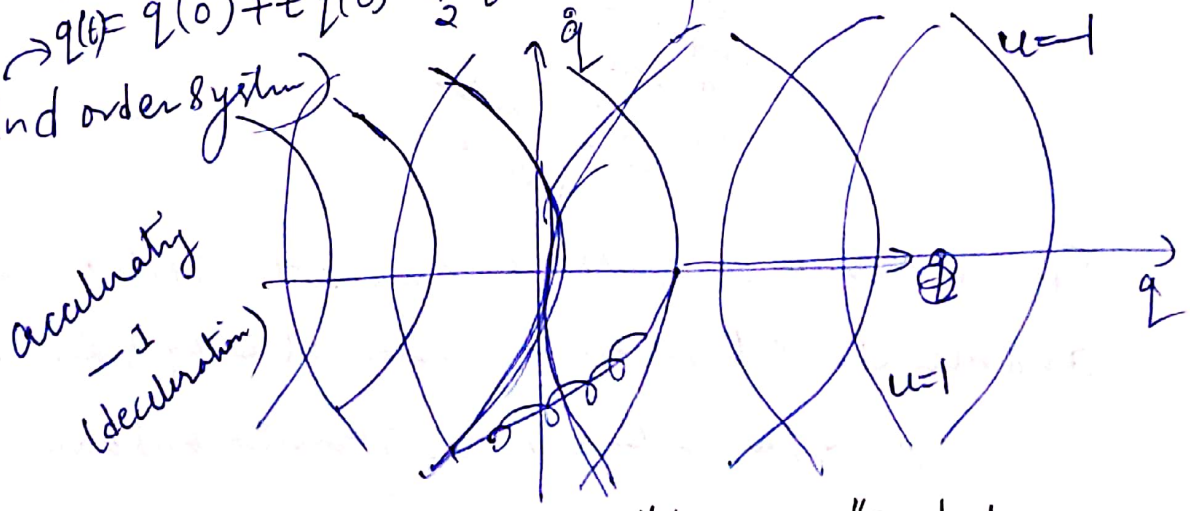
(2nd order system)

Steps

(1) What might the optimal policy be?

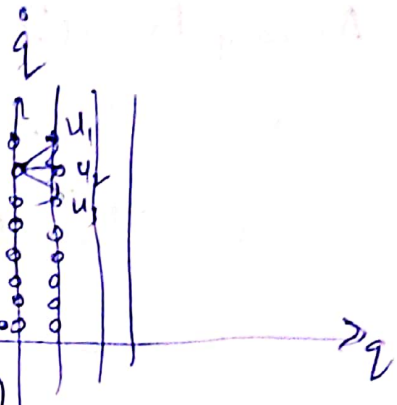
(II) How to write the cost function to get it?

(III) An algorithm to get q to 0.




"bang bang" control

# Dynamic programming




discrete states $s_i \in S$

discrete actions $a_i \in A$

discrete time dynamics

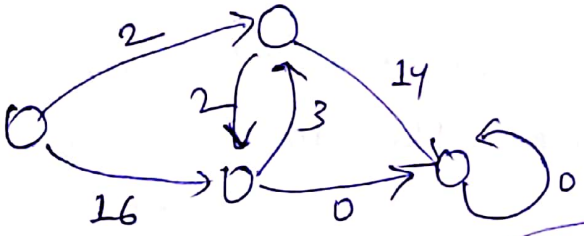
$$s[n+1] = f(s[n], a[n])$$

one step cost $l(s,a)$ ~ edge weight on the graph

total cost along some trajectory $\sum_{n=0}^{N} l(s[n], a[n])$

(accumulation of one-step cost)

(Additive cost)




in RL as well as control

## Cost-to-go function (aka Value function)

- optimal cost-to-go function

$$J^*(s_i) = \min_{a[i]} \sum_{n=0}^{N} l(s[n], a[n])$$

edge cost / transition cost.

$J^*(s_i)$: initial condition state $s_i$

$\min_{a[i]}$: min over a long term action (all of the actions into the future)

Subject to $s[0] = s_i$

$s[n+1] = f(s[n], a[n])$

If we start at state $s_i$ and do the best we can and take the best possible action at every state, how much total cost we are going to obtain.

The reason it is so useful because we can solve for it recursively.

$J^*$ ← notation for optimal