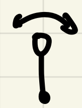
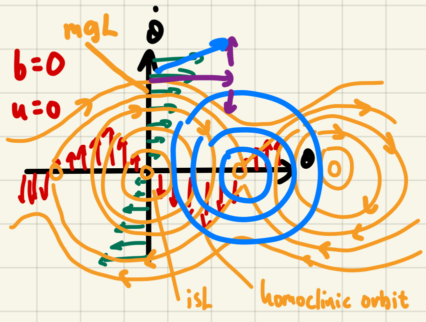
Lec 3

Change vector field by adjusting it to do your will
One idea: feedback linearization/cancellation

Phase Portrait – Undamped pendulum: $ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u$





fixed pt around upright pendulum position

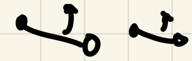
controller: $u = 2mgl\sin\theta$ (reverse gravity)

With sufficient torque

peak torque needed: $2mgl$

if insufficient torque: $u = \text{sat}(2mgl\sin\theta, -1, +1)$

will get stuck trying to get to top if not enough torque



can't arbitrarily move vector field

pt: $\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix}$

vector magnitude: $\begin{bmatrix}\dot{\theta} \\ \ddot{\theta}\end{bmatrix}$ control actions affect $\ddot{\theta}$, can't impact $\dot{\theta}$

only change y-dir

phase portrait only go CW in 2nd order system

How to change/rewrite vector to have new stable fixed pt? NONTRIVIAL

By rank constraint (1 DOF, 1 actuator) fine, but saturation limited so underactuated

with small torques need to pump up energy to get potential to reach top and regulate

Control as Optimization

Specify control problem as an optimization problem. optimization theory and numerical optimization

Given trajectory $x(\cdot), u(\cdot)$ shorthand for $\forall t, x(t), t \in [0, t_{max}]$

Assign score (1 scalar #)

RL: optimize reward, positive reinforcement

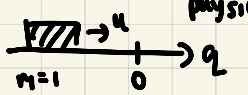
control theorist: minimize cost, penalization

Ex: time to goal, avg distance to trajectory

Many optimization formulations apply constraints. Only considered limited trajectories that satisfy these constraints. Find best one according to score
E.g. $|u| \le 1$ (torque limit)
$x(t_f) = x_{goal}$ (reaches goal state at final time)
Subtleties in cost specifications. What cost function to teach tying shoe or making salad?

Ex: Min line for double integrator

$\ddot{q} = u \quad |u| \le 1$



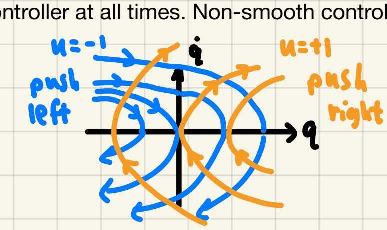
physical representation

goal: drive to $q = \dot{q} = 0$ in min time (from initial condition)

Optimal solution: max control input, then slam on breaks when necessary
"Bang-bang" policy: slamming on limits of controller at all times. Non-smooth controller

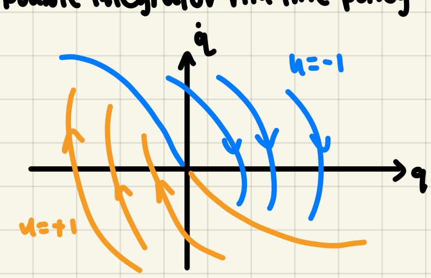
$\ddot{q} = u$

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



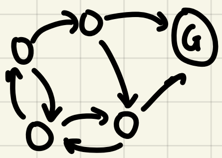
$\Rightarrow$

Double integrator min time policy

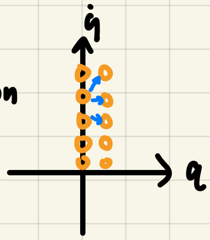


Lec 3 Minimum-time is like shortest path problem

graph search



discretization →



edges: taking actions from goal

Dynamic Programming (Recursive algo)

Discrete

Discrete states $s_i \in S$

Discrete actions $a_i \in A$

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

"edge cost" $g(s,a)$ total cost $\sum_n g(s,a)$

key idea: accumulation of simple costs along trajectory (additive cost) gives recursive structure

time to goal $g(s,a) = \begin{cases} 1 & \text{if } s \neq s_{goal} \\ 0 & \text{otherwise} \end{cases}$

solve backwards from the goal

$$J^*(s_i) = \underbrace{\min_{a[\cdot]}}_{\text{hard to search}} \sum_{n=0}^{\infty \leftarrow \text{watch for convergence}} g(s[n], a[n]), \; s[0] = s \implies J^*(s_i) = \min_a \left[\underbrace{g(s,a)}_{n=0} + \underbrace{J^*(f(s,a))}_{n: 1 \to \infty} \right]$$

↑ cost-to-go OR "value function" in RL

condition to certify optimality

Optimality certifier/checker

Trajectory is correct if cost-to-go satisfies self consistency condition (one step back from J leads to previous control action, satisfying the above equation)

Policy is good/controller is optimal if for every trajectory controller takes, its Cost-to-go certifiably meets the criteria of the optimality function where cost is minimized

Can also be turned into optimal trajectory algo

Other graph search algorithms can compute from a specified initial condition the optimal path to the goal
However, DP is the relevant paradigm as it computes the complete policy as a whole (finds control scheme so that it can formulate optimal trajectory from arbitrary initial condition to goal), and translates directly to continuous time formulation

Algorithm

$\hat{J}^* \leftarrow$ Estimate of optimal cost-to-go

$\forall i \;\; \hat{J}(s_i) \Leftarrow \min_a \left[g(s,a) + \hat{J}^*(f(s,a)) \right]$

$\hat{J}^* \to J^* + c$ (converges to optimal cost-to-go)

"dynamic programming" ↙ finite time

"value iteration" ← $t \to \infty$ (infinite horizon)

continuous

$\int g(s,a)\, dt$

Contraction metric that says that you can abuse it and still find optimal solution. You can pick random state and update and still find optimal policy

Caveats
Accuracy (discretization errors), systemic. Especially bad for solutions with discontinuities such bang-bang policy
Scalability (works only if can make fine enough mesh in state space, dimensions ~ 5)
- bellman curse of dimensionality

Model is known
Need a cost function (can't solve problems where there's arbitrary cost evaluation such as tying shoes or making salad)
Assumes "full state" feedback. To use controller, needs to know exact current state (we have model class with partial observable version of this problem but don't have satisfying solution) BIGGEST PROBLE

Complex dynamics but few dimensions, can be solved well even with uncertainty
Arbitrary number of dimensions but linear, can be solved well
All complex problems are intermediary (semi-complex dynamics, semi-large number of dimensions)