Lec 1 q = joint angles
u = input

Feedback cancellation

Given $\ddot{q} = f_1(q,\dot{q}) + f_2(q,\dot{q})u$
Given $\ddot{q}^d$ (desired accel)
then $u = f_2(q,\dot{q})[\ddot{q}^d - f_1(q,\dot{q})]$
$\Rightarrow \ddot{q} = \ddot{q}^d$

can think of system dynamics as
Feedback equivalent to $\ddot{q} = u$ (double integrator, has optimal solns)
still must take actions over time to origin
but know everything about controller

Robotics did this for 50 yrs, but f_2^{-1} may be power hungry/require high torques

"Erase" dynamics, impose w/ potentially lots of torque diff dynamics. partially limited by motors & control philosophy

Now, using stronger optimization tools to break out of current mold

Walking robots not fully actuated unless in the case of having big flat foot that's attached to ground and pretend no DOF btwn foot and ground that constrains motion robot can take. Now you act like fully actuated. In this regime you can make yourself a clockwork man. Such that we can think of robot as one big robot arm bolted to ground even when one leg comes off ground (as one leg is always bolted to ground)

Things that break feedback equivalence: (make robotics interesting, have to think of long-term consequence of actions)
W/ input saturation (controller demands torques that can't be produced)
State constraints (i.e. inequality constraint robot hand can't be inside table)
Model Uncertainty

Input saturation by generalized definition of under-actuated would make system under-actuated $u \in [-10, 10]$
State constraints (holonomic, non-holonomic)

Subtlety in definitions: Put very large torques and have no limit on bandwidth in case of no coupling (hack the parameters), can effectively make underactuated system look almost fully actuated

Some gym environments lost essence of dynamics they were intended to model to make learning curve look better

Study of walking robot =study of actuated robots (unless any portion of robot bolted to ground, then fully actuated)
Config to describe location of body in space
Humanoid is under-actuated even though more tendons and muscle tissue (dim of u) than joints (dim of q). Because can't immediately control equations of motion of center of mass. As soon as i jump into air, going to take ballistic trajectory excluding aero effects. Dim m motors can't control degrees of freedom.

Drake has symbolic engine that exposes symbolic structure/derivations of equations for certain algos that need it

Manipulator Equations for rigid body mechanics

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \tau_g(q) + Bu$$

← actuation selector
torque input
mass matrix (inertial matrix)
Coriolis terms
gravity terms

Exception: if using quaternions use diff notation

$M(q) > 0$ mass always pos (positive definite)

$T = \frac{1}{2}\dot{q}^T M(q)\dot{q}$ ($\frac{1}{2}mv^2$ in matrix form)

$\ddot{q} = M^{-1}(q)[-C(q)\dot{q} + \tau_g(q) + Bu] + \tau_{friction}$

$f_2 = M^{-1}(q)B$

if M known full rank & reversible then we only need to ask if B is full row rank (underactuated or not)

fully actuated - B is identity matrix
actuator for every motor
if low rank can't do feedback linearization

troody Pete leg lab

HONDA P2, P3, Asimo → heavily actuated
passive dynamic walkers → vs natural dynamics

Moral of story: push limits of natural dynamics with minimal control
Algebraicly slightly different from full actuation but very different in rollouts

Hierarchy of controllers
Some are making that assumption of zero moment point true, rest leverage that assumption
Also requires biggest moment are at ankle

Start of leg lab was dynamic robots (Marc Raibert founder of Boston dynamics) before HONDA. Running dyanamics easier than walking as you could throw yourself in air and do intermittent control

ATLAS- exploiting dynamics, writing optimizations to leverage dynamics and not cancelling them out
A scientific challenge. Humans have motor control systems that solve this problem but we don't know how to solve it as engineers
RL to get to limits of performance

Lec 1

NOT JUST LEGS
Moral equivalent to feedback linearization in aircraft/drones is staying in low angle of attack and airflow stays attached to wing, then have considerable control authority. Safe zone. Flaps/aelerons have significant authority of over pitch and the like

Birds don't restrict themselves to this small envelope. For example when they go on a perch. They go into severe post-stall maneuvers.
Clear airflow separation. Air not attached to wings. Go into stall case, but still land
Separated flow, lose control

Rock placed in front of trout in water tunnel. Rock sheds vortices. Trout adapts gait accordingly, called von karman gait
Trout is dead. It can swim upstream since mechanics of body designed to resonate with vortices that it experiences in the world turn the energy in the vortex straits into forward propulsion with no intelligence just dynamics
Dynamics is beautiful and you should master it not cancel it out

Machine learning motivation: Using perception to operate close to limits of vehicle

Lec 2

Steven Strogatz - uses graphical way to explain dynamical systems
Relevant for robots, learning theory
Long-term behavior of complicated robotics system, gonna need dynamics

Way people optimize performance of gradient descent algorithms is very similar to actuation optimization tools

Simple pendulum (all robots are connected pendulums)
walking systems (double pendulum is almost like one of our simplest walking models)

Kinetic energy
$T = \frac{1}{2} ml^2 \dot{\theta}^2$ $\left(\frac{1}{2}mv^2\right)$
Potential
$U = -mgl\cos\theta$

lagrangian mechanics
$ml^2\ddot{\theta} + mgl\sin\theta = Q$ ← generalized force (torque around joint in this case)

$Q =$ simplest friction model around joint (linear damping) + additional torque $\quad (Q = -b\dot{\theta} + u)$

$ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u$ ← adds non-linearity

$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \tau_g(q) + Bu$

given $\dot{\theta}, \theta, u$ → get $\ddot{\theta}$
$\ddot{\theta} = f(\theta, \dot{\theta}, u)$

Given: $\theta(0), \dot{\theta}(0)$ (I.C.)
You tell me: $\theta(t)$ with diff eq

But can't solve nonlinear diff eq (no close form expression)
→ no damping still get elliptic integrals
w/ damping, get nothing
(can get numerical approximation w/ simulator?
with numerical soln

Another Approach

(control theory hard bc long term consequence of action)

can't answer "where at time t". There are other analytically precise questions to ask
where at time ∞ → easier questions
If it starts at some place will it visit other place
time bad, other variables can be determined precisely
Long-term behavior, stability ($\lim t \to \infty$, easier than details to get there)
What is $\lim_{t\to\infty} \theta(t)$?
Will robot fall down?

Lec 2

Summarize robot by examining some characteristic behaviors of system and get very simple concept out
Simplest dynamics ideas can be more powerful than representation learning, machine learning, and other tools

Graphical Analysis (Steve Strogatz)

$ml^2\ddot{\theta} + \underbrace{b\dot{\theta}}_{\text{damping}} + mgl\sin\theta = u$

Theoretical physicists transformed robots beautiful things to say about how systems walked

heavily damped regime, simplifies 2nd to 1st order system $(b\dot{\theta} \gg ml^2\ddot{\theta})$ units N·m

$b\dot{\theta}$: $kg\frac{m^2}{s}$ $\quad$ ml^2: $kg\,m^2 \frac{1}{s^2}$ $\quad$ $kg\frac{m^2}{s^2}$

$\sqrt{\frac{g}{l}}\ [\frac{1}{s}]$ $\quad$ $b \gg \cancel{m}l^2$

$b\sqrt{\frac{l}{g}} \gg ml^2$ heavily damped regime

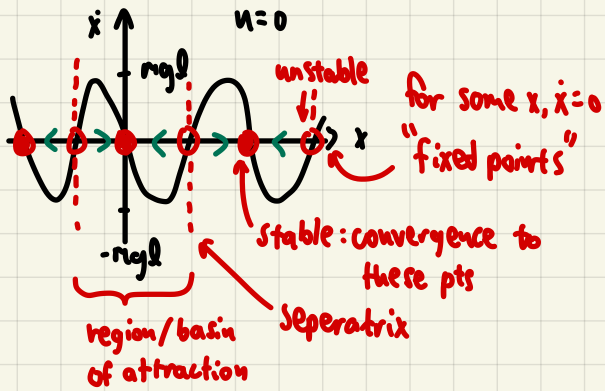
Tack on natural frequency term to reach dimensional parity

1st order

$b\dot{\theta} \approx u - mgl\sin\theta$

$b\dot{x} = u - mgl\sin x \leftarrow x \in \mathbb{R}$ (x no wrapping restriction of angles)

$u = 0$



linear: stable at origin no matter what
non-linear: wierd behavior
glancing contact w/ origin at pts like sin wave

Defn of Stability

global stability: all I.C.s converge to a pt

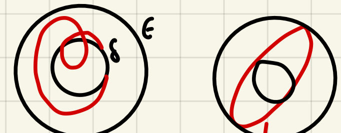
ϵ, δ are small pos constants

"local" stability

- In the sense of Lyapunov (i.s.L.): start at region, won't get too far

for every ϵ, $\exists\delta$ (there exists) s.t. $\underbrace{\|x(0) - x^*\| < \delta}_{\text{within } \delta \text{ ball of fixed pt in state space}}$ (x^*: f.p. (fixed pt)) $\Rightarrow \forall t$ (for all t) $\|x(t) - x^*\| < \epsilon$

if true for all t, for it to be true at x(0) then $\delta < \epsilon$



- locally attractive: will converge to region

- Asymptotically stable: attractive and isL $\quad \lim_{t\to\infty} x(t) \Rightarrow x^*$

- Exponentially stable: get to stable pt faster than linear system w/ particular const

$\forall t,\ \|x(t) - x^*\| < Ce^{-at}$ where $C, a > 0$

implies others

also, linear systems exponentially stable if stable

Craziness: # of fixed pts change w/ param change

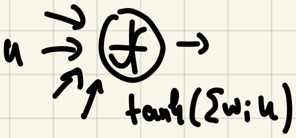
fixed pts come together / explode apart

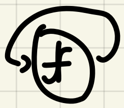
limit cycles, manifold stability

Simple Recurrent Neural Network

growing dendritic processes
simple illustrations of short-term memory

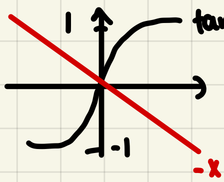
Short-term unit



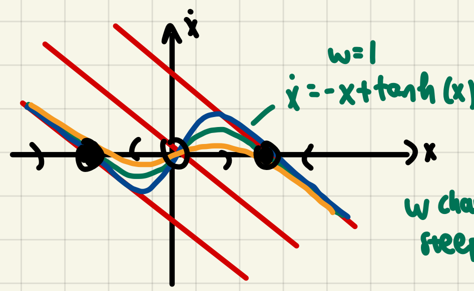


$\dot{x} = -x + \tanh(wx)$

Autapse (not known if it exists in brain but is found in a dish when cultured neurons are lonely & connect to themselves)



$\Rightarrow$

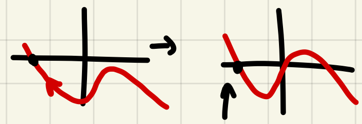


w changes gain of fn
steep or less steep

SIMPLEST analogy of short-term memory

"latching" mechanism with 2 fixed pts
bistable in off config

Can turn on to latch on one side
or turn off to latch to other side



stays here
when reset to off

Will remember it was
on that side

Long short-term memory (LSTM) - standard unit in RNN until transformers killed it
works on this principle

JANET (autapse with forgetting gate and more I/O)

Neuron = circuits.
Bistability = transistors
Direct analogs to analog electronics

Transformers also have a good dynamical systems interpretation if you use causal version of transformer
Dynamical systems theory and neural network crossover

- Rates of convergence, convergence (inequalities)
- Convergence not just to a fixed point. Optimization in neural network for instance, all minima are global minima. A bunch of fixed points are equally good

Hopfield network (hopfield's model of associate learning)- a recurrent neural network can store memories. Have multiple fixed points. Each one associated with different memory. Simple recipe to program a recurrent network to have fixed networks to be exactly where you want

DYNAMICS (time step in neural nets)

Set weights of neural network in simple case
Images are fixed points
Region of attraction: if close in some pixel space, will converge back to fixed points (images)

Function approximator paradigm being used to approach neural net, don't think about dynamics of learning enough. Memory happens in some flow. Time matters in way brain processes info. LLMs are dynamical system neural networks. Words build off previous words, sequence to sequence. Not flow beautifully like dynamical system would, but has the sense that the current state of the current thing has something about the state of the whole sentence

Recurrent networks didn't scale well. Longer the sentence, hit memory cap. Transformers take over. More neurons doesn't provide arbitrarily large amount of memory capacity

Lec 2

2nd order

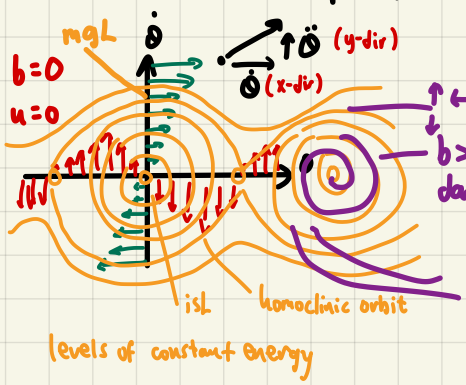
$$ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u$$

$\dot{x} = f(x,u)$ $x = \begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix}$

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \frac{1}{ml}[u - b\dot{\theta} - mgl\sin\theta] \end{bmatrix} \begin{matrix} \leftarrow x \\ \leftarrow y \end{matrix}$$

convert 2nd order to 1st order
write 2 eqns

Phase Portrait - Undamped pendulum



torque limited
impact local output vector magnitude
at time step

b>0
damping

underactuate may mean no
ability to change vector of a given
time

bands of I.C.s that
land at fixed pt

Fully actuated:
impose vector field
wiping field from natural
dynamics

Plot on top of this results of solving for value functions. Physics of problems will be revealed by optimal control problem

Control : Change the Vector Field (if u nonzero / a function)

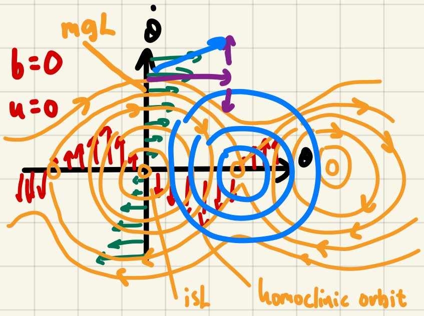
Game we want to play: what's the minimal change in vector field that shapes the dynamics, reflows the dynamics to the place we want

Lec 3

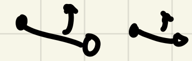
Change vector field by adjusting it to do your will
One idea: feedback linearization/cancellation

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

With sufficient torque
peak torque needed: $2mgl$



fixed pt around upright pendulum position
controller: $u = 2mgl\sin\theta$ (reverse gravity)
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

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

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

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

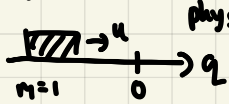
Ex: time to goal, avg distance to trajectory

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 time for double integrator

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

physical representation



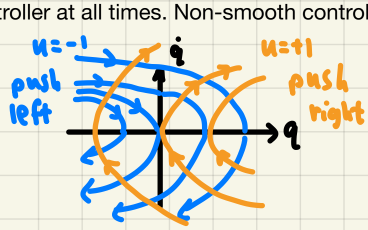
goal: drive to $q = \dot{q} = 0$ in min time
(from initial condition)

Double integrator min time policy

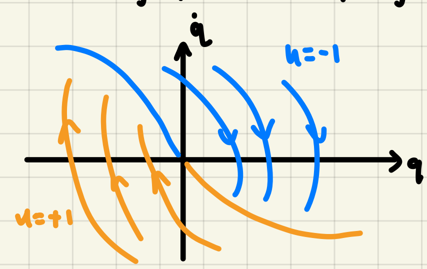
Optimal solution: max control input, then slam on breaks when necessary
"Bang-bang" policy: slamming on limits of controller at all times. Non-smooth controller

$\ddot{q} = u$

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

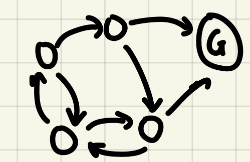


$\Rightarrow$

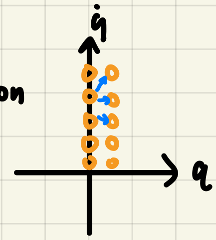


Lec 3 Minimum-time is like shortest path problem

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

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

↑ 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

continous

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

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

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

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

"dynamic programming" ↙ finite time

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

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)