Atlas : mechanics + optimization
Angel : RL-based
manipulation : Imitation-learning

△ underactuated

$\begin{cases} \dot{x} = f(x,u) \\ y = g(x) \end{cases}$ $\qquad \begin{cases} \dot{x} = Ax+Bu \\ y = Cx+Du \end{cases}$

△ second-order nonlinear system

$\ddot{q} = f(q,\dot{q},u)$

or

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

△ "control affine" nonlinear systems

$\ddot{q} = f_1(q,\dot{q}) + f_2(q,\dot{q})u \quad (1)$

based on this, define underactuated

def

(1) is fully actuated in $q, \dot{q}$ iff

$f_2(q,\dot{q})$ is full row rank (matrix)

i.e., $\dim(q) = m$, $\dim(u) = n$, $f_2(q,\dot{q}) \in \mathbb{R}^{m\times n}$, $\operatorname{rank}[f_2(q,\dot{q})] = m$

controllable : can choose some inputs over the to achieve some goal

actuable : can choose some inputs to cause acceleration

(1) is underactuated iff in $q, \dot{q}$ $\operatorname{rank}[f_2(q,\dot{q})] < m$

$\forall q, \dot{q}$, $\operatorname{rank}[f_2(q,\dot{q})] < m$

⇔ "system" is underactuated

△ Feedback equivalence (fully-actuated)

given $\ddot{q} = f_1(q,\dot{q}) + f_2(q,\dot{q})u$

$\ddot{q}_d$

⇒ then $u = f_2^{-1}(q,\dot{q})[\ddot{q}_d - f_1(q,\dot{q})]$

⇒ $\ddot{q} = \ddot{q}_d$

give me an "acceleration", I give u an "u"

⇒ feedback equivalent to $\ddot{q} = u$ (double integrator which we know optimal ctrl fn)

△ feedback equivalence be broken:
input saturations
state constraints
model uncertainty

△ Manipulator Eqs

(coriolis / Centrifugal Matrix)

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

mass matrix, coriolis terms, gravity, torque input

$M \succ 0$

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

$\ddot{q} = f_1(q,\dot{q}) + f_2(q,\dot{q})u$ is in this form!

d! whether $f_2(q,\dot{q})$ is full rank relies on "B"

as $\underbrace{M^{-1}(q)}_{\text{full rank}}\underbrace{B}_{\text{not sure!}}$

△ Dynamic Programming

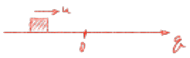
$ml\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u$

△ Control as an optimization

- given trajectory $x(\cdot), u(\cdot)$ — $\forall t \; x(t) \quad t \in [0, T_{max}]$
- Assign a score (scalar reward cost)
  e.g. time, avg distance
- subject to constraints

e.g. minimum-time for double integrator

$\ddot{q} = u$



goal : drive → $q = \dot{q} = 0$ in minimum-time from any initial condition

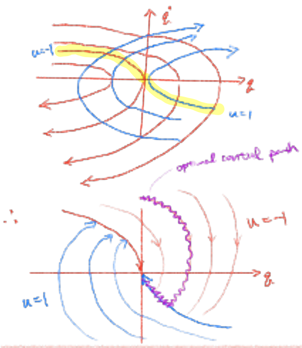
△ intuition: "bang-bang" policy (accelerate as much as possible) (unsmooth)

△ intuition #2:

$\ddot{q} = u \quad u = -1 \qquad u = 1$

$\dot{q}(t) = \dot{q}(0) - t$

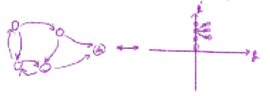
$q(t) = q(0) + \dot{q}(0)t + \frac{1}{2}at^2$



Q: How to generalize ?

△ Dynamic Programming

minimum-time ≅ shortest path problems



- weighted shortest path
- DP
  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 $\sum g(s,a)$
  Key idea : Additive cost
  e.g. time-to goal $g(s,a) = \begin{cases} 1 & \forall s \neq s_{goal} \\ 0 & \text{o.w.} \end{cases}$
  $\int g(s,a)\,dt$

△ DP

- DP is a recursive algorithm
- solve backwards from the goal
- cost-to-go

$J^*(s_i) = \min_{a[\cdot]} \sum_{n=0}^{\infty} g(s[n], a[n])$ (watch for converge)

$s[0] = s_i$

hard to search

⇔ $J^*(s_i) = \min_a [g(s,a) + J^*(f(s,a))]$ (single)

condition to certify optimality

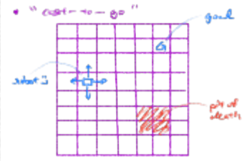
- Algorithm of DP (= value iteration)

$\hat{J}^*$ ← estimate of optimal cost-to-go

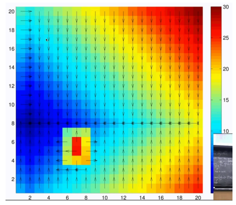
$\forall i \quad \hat{J}^*(s_i) \leftarrow \min_a [g(s,a) + \hat{J}^*(f(s,a))]$

$\hat{J}^* \longrightarrow J^* + c$

- "cost-to-go"



$g(s,a) = \begin{cases} 0 & @ \text{ goal} \\ 10 & @ \text{ pit} \\ 1 & @ \text{ o.w.} \end{cases}$



- Caveats
  - accuracy (discretization errors)
  - scalability (dimensions ~5) curse of the dimensionality
  - model is known
  - cost function
  - assume "full state" feedback

△ Lim DP ($\Delta x, \Delta u, \Delta t \to 0$)

- w/ minimum-time double integrator
  $\ddot{q} = u, \; |u| \le 1$
  optimal policy is "bang-bang"
- we want "discrete" → "continuous"

△ optimal cost-to-go

- discrete
  $s[n+1] = f(s[n], a[n])$
- cost
  $\sum \ell(s[n], a[n])$
  $\forall s \; J^*(s) = \min_a [\ell(s,a) + J^*(f(s,a))]$ (cost-to-go function)
  ∴ $\pi^*(s) = \arg\min_a [\ell(s,a) + J^*(f(s,a))]$
  (optimal controller/policy)

△ discrete v continuous

states	s	x
actions	a	u
time	n	t

$s[n+1] = f(s[n], a[n])$ — $\dot{x} = f_c(x,u)$

$\sum_{n=0}^{\infty} \ell_d(s[n], a[n])$ — $\int_0^{\infty} dt\, \ell_c(x,u)$

$\forall s \; J^*(s) = \min_a [\ell(s,a) + J^*(f(s,a))]$ — $\forall x \; 0 = \min_u [\ell_c(x,u) + \frac{\partial J^*}{\partial x} f_c(x,u)]$

$\pi^*(s) = \arg\min [\cdots]$ — $\pi^*(x) = \arg\min [\cdots]$

△ informal derivation

$x[n+1] \approx x[n] + h\, f_c(x[n], u[n])$

euler-integration $\quad t_{n+1} - t_n$

$\ell_d(x,u) = h\,\ell_c(x,u)$

∴ $J^*(x) = \min_u [h\,\ell_c(x,u) + J^*(x + h f_c(x,u))]$

$J^*(x) + \frac{\partial J^*}{\partial x} h f_c(x,u)$

⇒ $0 = \min_u [\ell_c(x,u) + \frac{\partial J^*}{\partial x} f_c(x,u)]$

Hamilton-Jacobi-Bellman Eqn.

HJB

e.g. infinite-fine Grid world

$J^*(x) = c$

$\frac{dJ}{dt} = \frac{\partial J}{\partial x} f(x,u)$

for u^* cost-to-go

$0 = \ell(x,u^*) + \frac{\partial J^*}{\partial x}\frac{\partial x}{\partial t} \Rightarrow \frac{\partial J}{\partial t} = -\ell(x,u^*)$ (running cost)

⇒ if you are following optimal policy your cost ↓

HJB sufficiency theorem

e.g. double integrator / Quadratic cost

$\ddot{q} = u \quad \ell(x,u) = q^2 + \dot{q}^2 + u^2$ (no constraints)

claim:

$\pi^*(x) = -q - \sqrt{3}\dot{q}$

$J^*(x) = \sqrt{3}q^2 + 2q\dot{q} + \sqrt{3}\dot{q}^2 \quad \left(x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}\right)$

proof :

(1) $0 = \min_u [q^2 + \dot{q}^2 + u^2 + \frac{\partial J}{\partial x} f(x,u)]$

$\left(\frac{\partial J}{\partial x} f(x,u) = \left[\frac{\partial J}{\partial q} \frac{\partial J}{\partial \dot{q}}\right]\begin{bmatrix} \dot{q} \\ u \end{bmatrix}\right)$

$= \min_u [q^2 + \dot{q}^2 + u^2 + \frac{\partial J}{\partial q}\dot{q} + \frac{\partial J}{\partial \dot{q}}u]$

$= \min_u [0]$

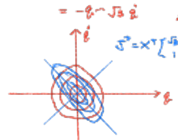
(2) $u = \arg\min_u [u^2 + \frac{\partial J}{\partial \dot{q}}u]$

⇒ $\frac{\partial}{\partial u}(\cdot) = 2u + \frac{\partial J}{\partial \dot{q}} = 0$

⇒ $u = -\frac{1}{2}\frac{\partial J}{\partial \dot{q}}$

$= -\frac{1}{2}(2q + 2\sqrt{3}\dot{q})$

$= -q - \sqrt{3}\dot{q}$



$J^* = x^T \begin{bmatrix} \sqrt{3} & 1 \\ 1 & \sqrt{3} \end{bmatrix} x$

△ LQR

- $\dot{x} = Ax + Bu$
  $\ell(x,u) = x^T Q x + u^T R u$
  $Q \succeq 0 \quad R \succ 0$
- inf horizon
  $J^*(x) = x^T S x$ (cost-to-go) can capture the dynamics
  $S \succeq 0$

$\frac{\partial J}{\partial x} = 2x^T S$

$0 = \min_u [x^T Q x + u^T R u + \frac{\partial J}{\partial x}(Ax + Bu)]$

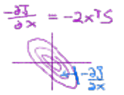
⇒ $\frac{\partial [\,]}{\partial u} = 0, \quad u = -R^{-1}B^T S x$
$= -Kx$

Riccati Equation

$0 = Q - SBR^{-1}B^T S + A^T S + SA$

- insights :

$-\frac{\partial J}{\partial x} = -2x^T S$ (where we want our cost-to-go func. to decrease)



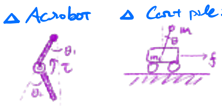
$u = -R^{-1}B^T S x$
(gradient)
(underactuated B matrix to adjust Sx)
(weighting on $B^T S x$)

△ DP so far

- Tabular (discrete → graph search)
- Linear dynamics + convex objective (LQR)

↳ here, use acrobot, cart pole, quadrotor to demonstrate DP

△ Acrobot △ Cart pole:



△ Equations of motion :

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

- Acrobot	- Cart pole
$q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$	$q = \begin{bmatrix} x_{cart} \\ \theta_{pole} \end{bmatrix}$
$u = [\tau]$	$u = [f_{cart}]$
$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
$\lvert u \rvert \le u_{max}$	$\lvert u \rvert \le u_{max}$
	$\lvert x \rvert \le x_{max}$

↳ target : swing up + balance

soln 1. tabular value iteration ?
yes, not elegant

2. LQR ?
Beautiful solution for balancing (need more for swing-up)

△ LQR for nonlinear systems

$\dot{x} = f(x,u) \Longleftrightarrow \dot{x} = Ax + Bu$

⇒ choose nominal x_0, u_0 (usually a fixed pt)

⇒ $\dot{x} \approx f(x_0, u_0) + \underbrace{\frac{\partial f}{\partial x}\Big|_{x=x_0, u=u_0}}_{A}(x - x_0) + \underbrace{\frac{\partial f}{\partial u}\Big|_{x=x_0, u=u_0}}_{B}(u - u_0)$

let

$\bar{x} = x - x_0$

$\dot{\bar{x}} = \dot{x} - \dot{x}_0$

$\bar{u} = u - u_0$

⇒ $\dot{\bar{x}} = A\bar{x} + B\bar{u}$

e.g. simple pendulum

- $ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = \tau$
  $x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \quad u = [\tau]$
- linearize about $\theta = \pi, \dot{\theta} = 0, u = 0$
- $\dot{x} = \begin{bmatrix} \dot{\theta} \\ \frac{1}{ml^2}[\tau - b\dot{\theta} - mgl\sin\theta] \end{bmatrix}$
- check fixed pt ✓
  $\dot{x}_0 = \begin{bmatrix} 0 \\ \frac{1}{ml^2}[0 - b\cdot 0 - mgl\sin\pi] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
- linearization

$\frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l}\cos\theta & -\frac{b}{ml^2} \end{bmatrix}\Bigg|_{\theta=\pi} = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & -\frac{b}{ml^2} \end{bmatrix}$

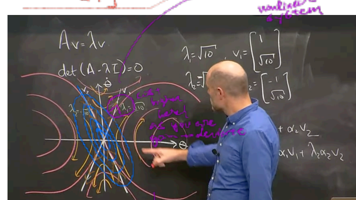
$\frac{\partial f}{\partial u} = \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix}$

let $m = l = 1 \quad g = 10$

$A = \begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix}$

$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

orange: approximate of a nonlinear system



red: nonlinear system

- LQR Design

$\begin{cases} \dot{\bar{x}} = A\bar{x} + B\bar{u} \\ J = \int_0^{\infty} \bar{x}^T Q \bar{x} + \bar{u}^T R \bar{u}\, dt \\ [K, S] = \mathrm{lqr}(A, B, Q, R) \\ u = -Kx = -R^{-1}B^T S x \\ J^* = \bar{x}^T S \bar{x} \end{cases}$

⇒ stable

$\dot{x} = \underbrace{(A - BK)}_{\text{Hurwitz}}x$

△ Is (A,B) controllable?

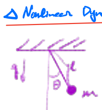
underactuated but controllable

△ controllable : given $x(0)$
can I find $u(t) \; \forall t \in [t_0, t_f]$
s.t. $x(t_f) = 0$

Is (A,B) stablizable

△ stablizable : given $x(0)$
can I find $u(t) \; \forall t \in [t_0, \infty)$
s.t. $\lim_{t\to\infty} x(t) = 0$
(no control input)

△ Nonlinear Dynamics



(kinetic energy)
$T = \frac{1}{2}ml^2\dot{\theta}^2 \quad v_{m} = l\dot{\theta}$
(potential energy)
$U = -mgl\cos\theta$

Lagrangian mechanics

$L = K - U$

$K = \frac{1}{2}mv^2 = \frac{1}{2}m\dot{s}^2 = \frac{1}{2}ml^2\dot{\theta}^2 \qquad \theta = \frac{s}{l}$

$U = mgy = -mgl\cos\theta$

∴ $L = \frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos\theta$

Euler-Lagrange equation

$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} = \frac{\partial L}{\partial \theta}$

$L = \frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos\theta$

$\frac{\partial L}{\partial \theta} = ml^2\ddot{\theta} + mgl\sin\theta = Q$ (generalized force)

$Q = \underbrace{-b\dot{\theta}}_{\text{friction}} + \underbrace{u}_{\text{torque}}$

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

⇕

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

⇕

$\ddot{q} = f(\theta, \dot{\theta}, u)$

⇕

$\theta(t) = ?$

△ Nonlinear Dynamics Questions

- What is $\lim_{t\to\infty}\theta(t)$?
- Will my robot fall down ?

$ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u \quad (N\cdot m)$

$kg\frac{m^2}{s^2}\frac{1}{s} \quad kg\,m^2\frac{1}{s^2} \qquad kg\frac{m^2}{s^2}$

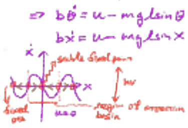
case : $b\dot{\theta} \gg ml^2$

$b\sqrt{\frac{l}{g}} \gg ml^2$ (heavily damped region)

⇒ linear approximation!

⇒ $b\dot{\theta} = u - mgl\sin\theta$

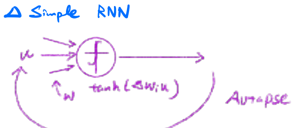
$b\dot{x} = u - mgl\sin x \quad x \in \mathbb{R}$



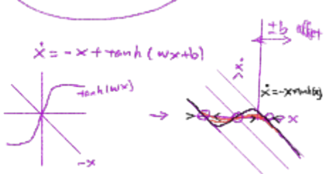
△ Stability

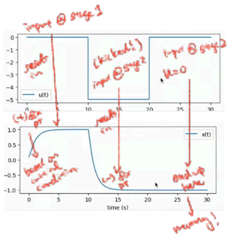
- local stability (do not leave a ball)
  - in the sense of Lyapunov (i.s.L.)
    $\begin{cases} \epsilon, \delta \text{: small positive constants} \\ \forall \epsilon, \exists \delta \text{ s.t. } \|x(0) - x^*\| < \delta \\ \quad \Rightarrow \forall t \; \|x(t) - x^*\| < \epsilon \end{cases}$
  - locally attractive — get to the region
    $\{\lim_{t\to\infty} x(t) \longrightarrow x^*$
  - asymptotically stable — i.s.L + attractive
    {isL + attractive
  - exponentially stable — get to the region w/ rate
    $\begin{cases} \forall t \\ \|x(t) - x^*\| < Ce^{-\alpha t} \\ C, \alpha > 0 \end{cases}$

△ Simple RNN



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





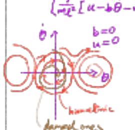
△ 2nd-order system

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

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

$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$

$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \frac{1}{ml^2}[u - b\dot{\theta} - mgl\sin\theta] \end{bmatrix}$



★ Essence of controls for underactuated robotics :

we hv all these dynamics

→ plug w/ u, s.t. phase portraits behave sth we want!!!

Δ Lyapunov Analysis

recall DP:
- Tabular — easy to explode
- LQR — only for linear case
- Approximate DP (NN) — works quite well (but solve a different DP, e.g. discounting)

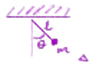
↳ all are trying to get "cost-to-go" function $J^*(x)$

(easy to find) (hard to find)

Δ now Lyapunov ⟷ optimal value
good enough — very good

might replace the original optimal ...

Δ Example: stability analysis of simple pendulum



$ml^2\ddot\theta + mgl\sin\theta = -b\dot\theta$

Δ $\theta(t)$ ⟶ hard to solve, cannot be analysis
Δ Lyapunov instead!!!

Δ $E = K + U$
$= \frac{1}{2}ml^2\dot\theta^2 - mgl\cos\theta$

Δ $\frac{d}{dt}E(x)$, $x = \begin{bmatrix}\theta\\ \dot\theta\end{bmatrix}$

$= \frac{\partial E}{\partial x}\frac{\partial x}{\partial t}$

$= \frac{\partial E}{\partial \theta}\dot\theta + \frac{\partial E}{\partial \dot\theta}\ddot\theta$

$= mgl\sin\theta\,\dot\theta + ml^2\dot\theta\ddot\theta$

$= \dot\theta(mgl\sin\theta + ml^2\ddot\theta)$

$= -b\dot\theta^2 \le 0$ if $b>0$

Δ General Energy function

Δ given $\dot x = f(x)$ no u
→ want to prove stability of $x^*=0$
→ construct a differentiable function $V(x)$, s.t.

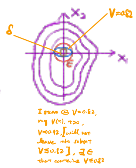
$\begin{cases} V(0)=0,\ V(x)>0,\ x\neq 0 & \text{PD}\\ \dot V(0)=0,\ \dot V(x)\le 0,\ x\neq 0 & \text{NSD}\end{cases}$

sufficient condition

→ then, x^* is stable i.s.L.

Δ i.s.L def

$\forall \epsilon>0,\ \exists\delta>0$
s.t. $\|x(0)-x^*\|<\delta$
$\forall t\ \|x(t)-x^*\|<\epsilon$



e.g. pendulum
$V(x) = E + mgl$

I start @ V=0.82, my V(x), t>0, V<0.82 [will not leave the subset V≤0.82], ∃ε that contains V≤0.82

Δ Asymptotically Stable

$\begin{cases} V(0)=0,\ V(x)>0,\ x\neq 0 & \text{PD}\\ \dot V(0)=0,\ \dot V(x)<0,\ x\neq 0 & \text{NSD}\end{cases}$

"<" not "≤"
otherwise $\dot V(x)=0$, ⟶ limit cycles.

Δ Global Stability

Global Asymptotically stability (GAS)

$\begin{cases} V(0)=0,\ V(x)>0,\ x\neq 0 & \text{PD}\\ \dot V(0)=0,\ \dot V(x)\le 0,\ x\neq 0 & \text{NSD}\end{cases}$

+

$\lim_{|x|\to\infty} V(x) = \infty$ "radially unbounded"

Δ Regional Stability

$\begin{cases} V(0)=0,\ V(x)>0,\ x\neq 0 & \text{PD}\\ \dot V(0)=0,\ \dot V(x)\le 0,\ x\neq 0 & \text{NSD}\end{cases}$

$\forall x\in D \subset \mathbb{R}^n$

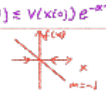
Δ Exponential Stability

$\begin{cases} V(0)=0,\ V(x)>0,\ x\neq 0 & \text{PD}\\ \dot V(0)=0,\ \dot V(x)\le -\alpha V(x),\ x\neq 0 & \text{NSD}\end{cases}$

⇕

$V(x(T)) \le V(x(0))e^{-\alpha t}$

e.g. $\dot x = -x$



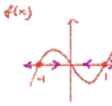
$V(x) = x^2$

$\dot V(x) = \frac{\partial V}{\partial x}\dot x$
$= 2x(-x)$
$= -2x^2 < 0$ { i.s.L ✓, asymptotic ✓
$\le 2V(x)$ { exponential ✓

$\lim_{|x|\to\infty} V(x) = \infty$ { global stable

e.g. $\dot x = -x + x^3 = f(x)$

{ $x^*=0$ is f.p.
$x\in(-1,1)$ R.O.A.



$V(x) = x^2$
$\dot V(x) = 2x(-x+x^3)$
$= 2x^2(x^2-1)$
$= \begin{cases} 0 & x=0,\ +1,\ -1\\ >0 & |x|>1\\ <0 & |x|<1,\ x\neq 0\end{cases}$

• sublevel set of V ⟶ invariant set
$V(x)\le\rho$ (inside the R.O.A. of x^*)

Δ General form of R.O.A.

if $V(x)>0$, $\dot V(x)<0$
$\forall x\in\{x \mid V(x)<\rho,\ \rho>0\}$ (invariant set)
then $V(x(0))<\rho$
⇔ $\lim_{t\to\infty} V\to 0$, $x\to 0$
and $\{x \mid V(x)<\rho\}$ is inside R.O.A.

Δ LaSalle's Theorem

Δ Lyapunov ⟶ Cost-to-go func.

HJB:
$0 = \min_u[\ell(x,u) + \frac{\partial J^*}{\partial x}f(x,u)]$
$u^* = \pi^*(x)$
⇒ $0 = \ell(x,u^*) + \frac{\partial J^*}{\partial x}f(x,u^*)$
$= \ell(x,u^*) + \frac{\partial J^*}{\partial t}$
⇒ $\dot J^*(x) = -\ell(x,u^*)$

"cost-to-go" function's derivative has to be decreasing!!!

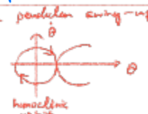
⇕ relaxation

$\dot V(x) < 0$ ⟶ way more easy!!!

HJB discrete
$J^*(x) = \min_u[\ell(x,u) + J^*(f(x,u))]$
$\Delta J^*(x) = J^*(f(x,u)) - J^*(x)$
⇕ $= -\ell(x,u^*)$
$\Delta V(x) \le 0$

Δ Lyapunov-based controller

e.g. pendulum swing-up



homoclinic orbit

• $E^d = mgl$
$V(x) = \frac{1}{2}(E(x) - E^d(x))^2$
$= \frac{1}{2}\tilde E^2(x)$

• $ml^2\ddot\theta + mgl\sin\theta = u$
$\dot E = u\dot\theta$
$\dot{\tilde E} = \dot E - \dot E^d = u\dot\theta$

• let $u = -k\dot\theta\tilde E$, $k>0$
making
$\dot{\tilde E} = -k\dot\theta^2\tilde E$

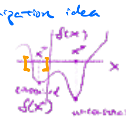
Δ Promise: find V(x) should be easier

- Global Lyapunov analysis for the pendulum
- input: pendulum dynamics
  parametric family of trig/poly for Lyapunov — polynomial over 1. cosθ, sinθ, θ̇
  output: coefficient of the polynomials
  + certificate that Lyapunov conditions satisfied ∀x ?

→ Lyapunov analysis w/ convex optimization

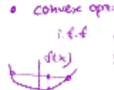
Δ some basic optimization idea

$\min_x f(x)$
s.t. $g_i(x)\le 0$



constrained $f(x^*)$ — unconstrained $f(x^*)$

• convex optimization
i.f.f. $f(x)$ is convex function
$\{g_i(x)\}$ is a convex set



LP → QP → SOCP ⟶ SDP
easy to solve ⟶ hard to solve

Δ SDP

linear objective, linear constraints
PSD matrix constraints.

Δ Robust stability analysis

$\dot x = Ax$, $A\in[\bar A, A^+]$ uncertain
⇒ $A = \sum_i\beta_i A_i$ $\beta_i\ge 0$, $\sum\beta_i = 1$
⇕
find $P\succ 0$ s.t. $\forall i$ $PA_i + A_i^TP \prec 0$
⇒ implies $P(\sum\beta_iA_i) + (\sum\beta_iA_i)^TP \prec 0$

↑

Δ $\dot x = Ax$ is it stable?

$V(x) = x^TPx$, $P\succ 0$
$\dot V(x) = 2x^TPAx$
$= x^TPAx + xA^TPx$ — Lyapunov fun linear system
$\dot V(x) < 0$ if $(PA + A^TP) \prec 0$
pick $Q\succ 0$ & $PA + A^TP = -Q$ (Lyapunov eq.)

find α s.t. $P = [\ldots]$
⇕ $P\succ 0$, $PA + A^TP \preceq 0$

find α,β $P\succ 0$ $P_2\succ 0$
s.t. $PA + A^TP = -P_2$

↑

Δ Q! How do we compute Lyapunov func?

1. Parameterize Lyapunov candidate
   let's say
   $\dot x = f(x)$ (linear NN)
   $V(x) = \sum_i\alpha_i\phi_i(x) = \alpha^T\phi(x)$ — nonlinear basis
2. Use convex optimization search over α to satisfy Lyapunov condition.

find α: $V(0)=0$, $\forall i\ V(x_i) > \epsilon x_i^Tx_i$
⇒ $V(x_i) > \epsilon x^Tx$ linear constraint in α
$\dot V(0) = 0$, $\forall i\ \dot V(x_i) = \frac{\partial V}{\partial x}\Big|_{x=x_i} f(x_i)$
$= \alpha^T\frac{\partial\phi}{\partial x}f(x_i)$
$< -\epsilon x_i^Tx$ — linear constraint

How do we make sure for all x?

↕

$\min_\alpha |\dot V(x) + 1|_1$

$V(x) = \sum_i\alpha_i\phi(x)$, $\alpha_i\ge 0$
★ $V(x) = \phi^T(x)G\phi(x)$, $G\succ 0$
$\dot V(x) = \frac{\partial V}{\partial x}f(x)$
⟶ construct $-\phi^T(x)G_2\phi(x)$
⟶ let $\dot V(x) = -\phi^TG_2\phi$

Δ DP sum

$\frac{d}{dt}x = f(x(t), u(t), t)dt$

• $J(x(t), u(t), T_0, T_f)$
$= Q(x(T_f), T_f) + \int_0^{T_f}\ell(x(\tau), u(\tau))d\tau$

• $V(x(T_0), T_0, T_f)$ (cost-to-go)
$= \min_{u(t)} J(x(t), u(t), T_0, T_f)$

• $-\frac{\partial V}{\partial t} = \min_{u(t)}\left(\left(\frac{\partial V}{\partial x}\right)^T f(x(t), u(t)) + \ell(x(t), u(t))\right)$

⇔ $\min_{u(t)}\left(\left(\frac{\partial J^*}{\partial x}\right)^T f(x(t), u(t)) + \ell(x(t), u(t))\right)$

Hamilton-Jacobi-Bellman (HJB)

• $V(x(T_0), T_0, T_f)$
$= V(x(T_0), T_0, t) + V(x(t), t, T_f)$

Bellman Optimality

⇔ $J^*(s_i) = \min_a[\ell(s_i,a) + \gamma J^*(f(s_i,a))]$

proof — total derivative

$\frac{d}{dt}V(x(t), t, T_f) = \frac{\partial V}{\partial t} + \left(\frac{\partial V}{\partial x}\right)^T\frac{\partial x}{\partial t}$

$= \frac{d}{dt}\min_{u(t)} J(x(t), u(t), t, T_f)$

$= \frac{d}{dt}\min_{u(t)} Q(x(T_f), T_f) + \int_t^{T_f}\ell(x(\tau), u(\tau))d\tau$

$= \min_{u(t)}\frac{d}{dt}\left[Q(x(T_f), T_f) + \int_t^{T_f}\ell(x(\tau), u(\tau))d\tau\right]$

$= \min_{u(t)}\ 0 + \frac{d}{dt}\left(\left[\mathcal{L}(x(\tau), u(\tau))\right]_t^{T_f}\right)$

$= \min_{u(t)}\frac{d}{dt}\left(\mathcal{L}(x(T_f), u(T_f)) - \mathcal{L}(x(t), u(t))\right)$

$= \min_{u(t)} -\ell(x(t), u(t))$

⇕

$\frac{\partial V}{\partial t} + \left(\frac{\partial V}{\partial x}\right)^T\frac{\partial x}{\partial t} = \min_{u(t)} -\ell(x(t), u(t))$

⇕

$-\frac{\partial V}{\partial t} = \min_{u(t)}\left(\left(\frac{\partial V}{\partial x}\right)^T\frac{\partial x}{\partial t} + \ell(x,u)\right)$

$= \min_{u(t)}\left(\frac{\partial V^T}{\partial x}f(x,u) + \ell(x,u)\right)$

★ Get V ⟶ Get u
- two point bounded value problem
- shooting method

Δ Discrete-Time HJB

• $x_{k+1} = F(x_k, u_k)$
• $J(x_0, \{u_k\}_{k=0}^n, n)$
$= Q(x_n, t_n) + \sum_{k=0}^n \ell(x_k, u_k)$
• $V(x_0, n)$
$= \min_{\{u_k\}_{k=0}^n} J(x_0, \{u_k\}_{k=0}^n, n)$
• $V(x_0, n)$
$= V(x_0, k) + V(x_k, n-k)\ \forall k\in(0, n$
• $V(x_k, n)$
$= \min_{u_k}\left[\ell(x_k, u_k) + V(x_{k+1}, n-1)\right]$, $x_{k+1} = F(x_k, u_k)$

⇔ vs $J^*(s_i) = \min_a[\ell(s_i,a) + J^*(f(s_i,a))]$

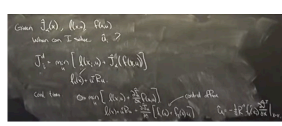
Discrete HJB

⇕

$V(x_k, n)$
$= \min_{u_k}\left(\ell(x_k, u_k) + V(F(x_k, u_k), n-1)\right)$

⇕

$V(x) = \min_u\left(\ell(x,u) + V(F(x,u))\right)$
↓
$\pi(x) = \arg\min_u\left(\ell(x,u) + V(F(x,u))\right)$



- use optimal u: recall
- instead of sampling u_i
  only sample $x_i\in X_s\subset X$

CT LQR: $u^* = -K^*x$, $= -R^{-1}B^TS^*x$
DT LQR: $u^* = -K^*x$, $= -(R + B^TSB)^{-1}B^TSAx$

⇒ $J_i^d = x_i^TQx_i + \hat u_i^TR\hat u_i + \hat J_\alpha(Ax_i + Bu_i)$
w/ optimal u w/ current $\hat S$

Δ DP (cont'd)

key idea
1) LQR (linear optimal control)
2) local linearization
3) $Q = \begin{bmatrix}10 & & & \\ & 10 & & \\ & & 1 & \\ & & & 1\end{bmatrix}$ $R = [1]$

@ balancing

Q: How do we swing up?
"value iteration"

⇒ Approximate DP

Δ Value iteration on a mesh for Pendulum Swing-up

Δ mesh q 51 bins — odd for "0"
$\dot q$ 51 bins
u 9 bins

Δ Acrobot
51^4 bins
9 bins ↗ explode!!!

Δ NN! Approximating Value Functions

$\hat J_\alpha(x)$
└ parameters + bias

- value iteration update
  • recall:
  $\forall s_i\in S$
  $\hat J(s_i) = \min_a\left[\ell(s_i,a) + \hat J(f(s_i,a))\right]$
  $s[n+1] = f(s[n], a[n])$
  $\sum_{n=0}^\infty \ell(s[n], a[n])$

  • Draw samples
  $x_i\in X_s\subset X$
  $u_j\in U_s\subset U$
  $\forall i,j$ $\dot x_{ij} = f(x_i, u_j)$ next state samples
  $\ell_{ij} = \ell(x_i, u_j)$
  desired J @ i
  $J_i^d = \min_j[\ell_{ij} + \hat J_\alpha(x'_{ij})]$

⇒ $\min_\alpha$ loss $\sum_i[\hat J_\alpha(x_i) - J_i^d]^2$
⇕
$\min_\alpha$ loss $\sum_i\left[\hat J_\alpha(x_i) - \min_j[\ell_{ij} + \hat J_\beta(x'_{ij})]\right]^2$
"target network"

"here we do the training based on the samples. To hv a better sampling, RL kicks in, & guide the state space, to a subset, that is closer to the optima."

Q: when does it work?
does it converge
↓ Δ linear approximator
Δ works @ special case: linear NN

• $\hat J_\alpha(x) = \alpha^T\phi(x)$ — nonlinear basis "input features" & fixed
$= \sum_j\alpha_j\phi_j(x)$

⇒ $\min_\alpha\sum_i[\alpha^T\phi(x) - J_i^d]^2$

linear least squares
has closed-form soln
⇒ SGD is guaranteed to converge to global minima

Δ Back to LQR; connection to approximator; w/ discrete LQR

$J^*(x) = x^TS^*x$
NN ⟶ linear approximator

loss $\sum_i[x_i^T\hat Sx_i - J_i^d]^2$

$\hat S \leftarrow \hat S - \eta\frac{\partial\, loss}{\partial S}$

Based on this, loss explodes↑

why? LQR tries to solve inf horizon, cost cannot go to inf.
How does normal LQR avoid inf?
$x\to 0$, $u\to 0$ (exact 0)
How about here, when we do sampling? popularly seen in RL

- discounting $\sum_{n=0}^\infty\gamma^n\ell(x[n], u[n])$
  $0\le\gamma\le 1$
  preferred that is more stable
  optimal: LQR($\sqrt{\gamma}A$, B, Q, $\frac{1}{\gamma}R$)
  $x[n+1] = Ax[n] + Bu[n]$
  $|eig(A)|\le 1$.

△ Let's continue on Lyapunov

Recall:

$V(x) > 0$

$\dot V(x) < 0$

- Linear neural network: $\forall x_i \to$ Linear Program
- Question: farm samples: $\forall x_i \to \forall x \in \mathbb{R}^n$

e.g. $\dot x = Ax$

$$\begin{cases} V(x) = x^T P x & \forall x \neq 0,\ P \succ 0 \\ \dot V(x) = x^T P A x + x^T A^T P x < 0 & (PA + A^T P) \prec 0 \end{cases}$$

SDP:

find $P \succeq 0$, $P_2 \succeq 0$ — Linear Matrix Inequality

$PA + A^T P = -P_2$ (linear constraints)

△ Sums of squares

- given a function $f(x)$, is $f(x) \geq 0\ \forall x$?

let $f(x) = \phi^T(x) P \phi(x)$, $P \succeq 0$

$= \sum \alpha_i \phi_i^2(x)$, $\alpha_i \geq 0$

sums of squares (SOS) — decomposition of the problem

then we can know $f(x) \geq 0\ \forall x$

- special case: Polynomials

$P_\alpha(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \dots$

$= \phi^T(x) P \phi(x)$, $P \succ 0$ — basis function

e.g. $p_\alpha(x) = x^T [\,]$ ... gets — $\phi(x) = [\,]$ expand $\phi(x)$

simplest basis: $m(x) \equiv$ monomial basis, e.g. $1, x, x^2, x^3, \dots$; $1, x, y, xy, x^2, y^2$

e.g. $2 - 4x + 5x^2$

$$= [1\ \ x] \begin{bmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix}$$

$\Downarrow$

find $P \succeq 0$

s.t. $P_{11} = 2$, $2P_{12} = -4$, $P_{22} = 5$

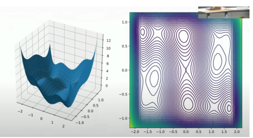
SDP again!

- linear objective
- linear constraints
- PSD constraints

SOS: $P_\alpha(x)$ is SOS → formulate SDP & solve → $\forall x \in \mathbb{R}^n$, $P_\alpha(x) \geq 0$

△ Six-hump camel

$$P(x, y) = 4x^2 + xy - 4y^2 - 2.1x^4 + 4y^4 + \frac{1}{3}x^6$$



SOS trick!

$\min_\gamma -\gamma$ s.t. $P(x,y) - \gamma$ is SOS

γ & P_2

?

$f(x) = NN(x)$, $NN(x)$

△ Back to Lyapunov

- Choose $V(x)$ to be polynomial
- assume $\dot x = f(x)$ restricted to polynomial

$\Rightarrow \dot V(x) = \frac{\partial V}{\partial x} f(x)$ is still polynomial

$$\begin{cases} V(x) > 0 \longrightarrow V(x) \text{ is SOS} \\ \dot V(x) < 0 \longrightarrow -\dot V(x) \text{ is SOS} \end{cases}$$

New constraints!

e.g.

$\dot x_1 = -x_1 - 2x_2^2$

$\dot x_2 = -x_2 - x_1 x_2 - 2x_2^3$

$V(x) = x_1^2 + 2x_2^2$, $\forall x$

$-\dot V(x)$ is S.O.S?

$\Rightarrow -\dot V(x) = m^T(x) P_x m(x)$ — decision variable

inversely: find Lyapunov function

$V(x) = (S_{2,2} + 2x_0 x_1 S_{1,0} + 2x_0 S_{2,1} + x_0^2 S_{1,1} + 2x_1 S_{2,0} + x_1^2 S_{0,0})$

Solution:

$V(x) = (1.000000x_0^2 + 2.000057x_1^2)$

△ R.O.A again

So basically:

- oracle answers T/F $\forall x \in \mathbb{R}^n$ $p(x) \geq 0$
- New Q: $\forall x \in D \subseteq \mathbb{R}^n$ is $p(x) \geq 0$

How:

define $D = \{x \mid g(x) \leq 0\}$ — polynomial

find α s.t. $p(x) + \lambda_\alpha(x) g(x)$ is SOS — adding decision variable

$\lambda_\alpha(x)$ is SOS

→ S-procedure — multiplier

- if $p(x) - \lambda g$ is SOS, $p(x)$ is also SOS
- if $g(x) \geq 0$, outside the region, $p(x)$ ↑

$p(x)$, $g(x) \leq 0 \Rightarrow p(x) \geq 0$

△ Trajectory Optimization

SO FAR:

- Dynamic Programming / optimal control
  - tabular (mesh-based)
  - LQR
  - NN (function approximator) (need sampling distributions)
- Lyapunov (simpler DP)
  - sums of squares opt. (10~20 dim) (more w/ SOS approximation)

for higher dimension?

$\forall x \longrightarrow$ NO

just $x[0] = x_0$ (1 initial condition)

→ motivation

△ Traj opt w/ linear discrete

Choose finite horizon N

$$\min_{x[\cdot], u[\cdot]} \sum_{n=0}^{N-1} \ell(x[n], u[n]) + \ell_f(x[N])$$ — terminal cost

over every $\forall n \in [0, N-1]$

$$\text{s.t.} \begin{cases} x[n+1] = Ax[n] + Bu[n] \\ x[0] = x_0 \\ -1 \leq u[n] \leq 1 \end{cases}$$

Choose $\ell(x,u) = x^T Q x + u^T R u$

but why? this is LQR! — QP (L-MPC)

- LQR v QP
  - inf horizon — finite horizon
  - no constraints — can add constraints!
- horizon $N \longrightarrow$ limiting? affecting LQR?
  - idea: choose $\ell_f(x) = x^T S x$ — LQR

$\sum_{n=0}^{N-1} \ell(x,u) + \ell_f(x)$ — $N \sim \inf$; constraints imposed on here

- can restrict your state!

$\min_{x[\cdot], u[\cdot]} \sum_{n=0}^{N-1} |u[n]|^2$

s.t. $x[0] = x_0$, $x[N] = x_f$

$x[n+1] = Ax[n] + Bu[n]$ $\forall n$

$|u[n]| \leq 1$

- min time?

$\ddot q = u$, $|u| \leq 1$

set N big → be feasible, then do $N-1$ recursively

△ Categories

- Direct transcription
  - decision variables: $x[\cdot]$, $u[\cdot]$
  - constraints: $x[n+1] = Ax[n] + Bu[n]$
- Direct "shooting"
  - decision variables: $x[0]$, $u[\cdot]$
  - constraints (linear): $x[n] = A^n x[0] + \dots$ — linear in $u[\cdot]$

△ MPC

- estimate current state $\hat x$
- solve linear traj opt w/ $x[0] = \hat x$
- execute $u[0]$
- repeat — "Recursive Feasibility"

★ when ever we are using it in feed, we are actually looking (w/ predefined by)

$x_f \longrightarrow 0$, where we are optimizing u w.r.t. error state.

△ Nonlinear Traj. Opt.

$x[n+1] = f(x[n], u[n])$ (possibly non-convex)

⇒ transcript / direct } could work: NOT CONVEX

SQP: 1. take big steps (convergence ...) (second-order method) 2. have to handle constraints

△ R.O.A. using SOS + S-procedure

e.g. $\dot x = -x + x^3$

$V(x) = x^2$

we know ROA is $\in [-1, 1]$

$\therefore V(x) \leq 1$

⇒ find α s.t. $-\dot V(x) + \lambda_\alpha(x)(1 - V(x))$ is SOS & $\lambda_\alpha(x)$ is SOS

$D = \{x \mid g(x) = 0\}$

find α: $p(x) + \lambda_\alpha^T(x) g(x)$ is SOS

$\lambda_\alpha(x)$ can be arbitrary

△ Traj. Opt.

last time

- direct transcription

$$\min_{x[\cdot], u[\cdot]} \sum_{n=0}^{N-1} \ell(x[n], u[n]) + \ell_f(x[N])$$

s.t. $x[n+1] = f(x[n], u[n])$

$x[0] = x_0$

$x \in X$

$u \in U$

- direct shooting

$$\min_{u[\cdot]} \sum_{n=0}^{N-1} \ell(x[n], u[n])$$

$x[n] = f(u)$

above in discrete time

continuous time

$\dot x = f_c(x, u)$ — e.g. $\min_{\text{search}}$ □ here: can be time decision variable

⇒ approximate w/ discrete time

⇒ $x[n+1] = x[n] + h f_c(x[n], u[n])$

Euler integration (can do RK!)

◁ when doing opt. → coarse h to speed up

△ Direct collocation

△ $\min_{x[\cdot], u[\cdot], h} \sum_{n=0}^{N-1} \ell_c(x[n], u[n]) \cdot h$

s.t. "dyn. constraints"

△ given $x[\text{samples}]$, $\dot x[\text{samples}]$

⇒ completely defines $x(t)$ as a cubic polynomial

⇒ $\dot x(t_{\text{collocation}})$

$= f_c(x(t_{\text{col}}), u(t_{\text{col}}))$

△ constraints

- I want gradients.
- signed distances (for collision avoid.)

$\phi_{ij}(q) \geq d_{\min}$ — pairs of bodies

$\min_{ij}[\phi_{ij}(q)] \geq d_{\min}$

1 constraint adding to other — in practice: softmin, softmax

hinge$(d) = 1$

softmin(hinge(ϕ_{ij}))

△ non-convex opt. in traj. opt. (must 3)

OPT? / OPT — convex set to opt.? big convex levels to local min.

△ Traj. Stabilization

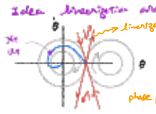
- we opt $x[\cdot], u[\cdot]$ || $x(t), u(t)$

the thing we got might not be able to execute, due to the optimization discretization.

- How do we stabilize?
  - MPC

• Traj. Opt.	• Controller.
• generalization of LQR → handle constraint	• generalization of LQR → stabilize

- Idea: linearization around traj.



linearization. phase portrait of pendulum

△ $\dot x = f(x, u)$

$\approx f(x_0, u_0) + \frac{\partial f}{\partial x}\Big|_{x_0, u_0}(x - x_0) + \frac{\partial f}{\partial u}\Big|_{x_0, u_0}(u - u_0)$

Now, this is not!!!

△ $x_0(t)$, $u_0(t)$

$\bar x = x - x_0(t)$, $\bar u = u - u_0(t)$ } time-varying coordinate system

△ $\therefore \dot x = f(x, u)$

$\approx f(x_0(t), u_0(t)) + \frac{\partial f}{\partial x}(x - x_0(t)) + \frac{\partial f}{\partial u}(u - u_0(t))$

⇒ $\dot x = \dot x_0(t) + A(x - x_0(t)) + B(u - u_0(t))$

⇒ $\dot{\bar x} = A(t)\bar x + B(t)\bar u$

⇒ optimal control for time-varying linear sys

e.g. finite-horizon LQR

△ recall

$\min_{u(t)} \int x^T(t) Q x(t) + u^T(t) R u(t)\, dt$

$\dot x(t) = Ax(t) + Bu(t)$

cost-to-go: $J(x, t) = x^T S(t) x$, $S(t) \succeq 0$

$u = -K(t)x$

where $S(t_f) = 0$

$-\dot S(t) = S(t)A + A^T S(t) - S(t)BR^{-1}B^T S(t) + Q$

Riccati diff. eq. — only diff: $= 0$

this is from HJB w/ $\min_u [\ell(x,u) + \frac{\partial J}{\partial x} f(x,u) + \frac{\partial J}{\partial t}] = 0$

△ A, B, Q, R → all can be time-varying

△ Therefore, when tracking a traj.

$\bar u = -K(t)\bar x$

⇒ $u - u_0 = -K(t)(x - x_0)$

⇒ $u = u_0 - K(t)(x - x_0)$

△ Sum

1. get a traj
2. approximate dynamics (linearization) (moving coordinate)
3. do optimal control
4. + constraints: generalized LQR to LMPC w/ constraints

△ Recall

- best the dimension (in Traj. Opt)
- solve, based on x_0, initial cond.

Traj. Opt

- direct transcription
- direct shooting
- direct collocation

} give $x(t)$, $u(t)$ "open-loop"

◁ problem: we cannot just use this $u(t)$ to our system

- we solve $u(t)$ w/ some solver. The "resolution" is definitely idealized w/ physics engine or real life
- "integration" different!

sol: linearize around the traj.

$x_0(t)$, $u_0(t)$ $\forall t \in [t_0, t_f]$

$A(t)$, $B(t)$

w/ Finite-horizon LQR, solve it w/ differential Riccati Eqn

gives $\bar u(t) = -K\bar x(t)$

$u(t) = u_0(t) - K(x - x_0(t))$

$J(\bar x, t) = \bar x^T S(t) \bar x$

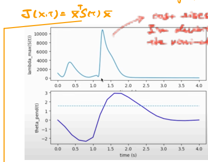
△ more on Finite-horizon LQR

e.g. cart-pole

$$\begin{cases} 2\ddot x + \ddot\theta\cos\theta - \dot\theta^2\sin\theta = f \\ \ddot x\cos\theta + \ddot\theta + \sin\theta = 0 \end{cases}$$

@ $\cos\theta = 0$, $\dot\theta$ doesn't really affects the angle. but w/ LQR, we can actually go through into the future, and be "lining up".

$J(x, t) = \bar x^T S(t) \bar x$ — cost rises, when I'm deviating from the nominal traj.



draw $S(t)$ in lower dimension

$J(x, t) = c$

- when $S(t)$ ↑, the level-set become relatively small
- to be successful @ t_f, I need to hit the small level set!

△ Traj. Funnels

- recall ROA: $V(x) \leq \rho \Rightarrow \dot V(x) \leq 0$, e.g. SOS

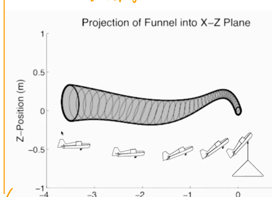
if proved ⇒ $V(x) \leq \rho$ is an invariant set.

- Traj. funnel: $V(x, t) \leq \rho \Rightarrow \dot V(x, t) \leq 0$ (this can also be time-varying $V(x,t) \leq \rho(t)$)

$\forall t \in [t_0, t_f]$

⇒ $V(x, t) \leq \rho$ is an invariant set $\forall t \in [t_0, t_f]$

we can then:
- reachability analysis
- do reprojection.



△ Time varying cost-to-go

when cost-to-go time-dependent

$J(x, t)$

$0 = \min_u [\ell(x,u) + \frac{d}{dt}J(x, t)]$

$= \min_u [\ell(x,u) + \frac{\partial J}{\partial x} f(x,u) + \frac{\partial J}{\partial t}]$

△ Time varying Lyapunov

- $\forall t$ $V(t, x) > 0$

⇒ $\forall t$ $\forall x \neq 0$ $V(t, x) > 0$, $V(t, 0) = 0$

when $\frac{d}{dt}J(x,t) = -\ell(x, u)$

- $\frac{d}{dt} V(t, x) = \frac{\partial V}{\partial x} f(x) + \frac{\partial V}{\partial t} \leq 0$

△ overall:

1. give $x_0(t)$, $u_0(t)$ (optimized traj.) — direct collocation, piecewise linear
2. time varying linearization — get $\dot{\bar x} = A\bar x + B\bar u$, $J(x, t) = \bar x^T S \bar x$ cost-to-go, $S(t)$, $K(t)$
3. proof the stability → the funnel of the optimized traj. "ROBUSTNESS" — tube MPC.

△ More $A(t), B(t)$

$x(t)$, $u(t)$

- major limitation: tries to "catch up" in time. error 一直在累积

△ 3 options for traj. stabilization

1) LQR (finite-horizon) — SOS verification / funnel
2) linear MPC (w/ linearization) — recursive feasibility
3) nonlinear MPC — few guarantee

MPC is traj opt → do stabilization; also: stabilization for traj opt

△ iterative LQR (iLQR) — closely related

differential dynamic programming — 2^{nd} order app. of $f(x,u)$; linearization of $f(x,u)$

idea: traj opt

$$\min_{x[\cdot], u[\cdot]} \sum_{n=0}^{N-1} \ell(x[n], u[n])$$

s.t. $x[n+1] = f(x[n], u[n])$, $x[0] = x_0$ — solve it by SQP (second-order approximation)

approximate by QP

$$\min_{x[\cdot], u[\cdot]} \sum_{n=0}^{N-1} \bar x^T Q \bar x + \bar u^T R \bar u + \text{const} \dots$$

s.t. $x[n+1] = A\bar x[n] + B\bar u[n] + c$, $x[0] = x_0$

this is LQR! generalized LQR problem!

△ Sum

1. I want to solve — initial; x_0, u_0 — 2. get LQR A, B; 3. repeat — x, u