Optimal Control for
Linear Dynamical Systems and Quadratic Cost
("LQR")

Pieter Abbeel
UC Berkeley EECS

Bellman's curse of dimensionality

- n-dimensional state space
- Number of states grows exponentially in n (assuming some fixed number of discretization levels per coordinate)
- In practice
  - Discretization is considered only computationally feasible up to 5 or 6 dimensional state spaces even when using
    - Variable resolution discretization
    - Highly optimized implementations

This Lecture

- Optimal Control for Linear Dynamical Systems and Quadratic Cost (aka LQ setting, or LQR setting)
  - Very special case: can solve continuous state-space optimal control problem exactly and only requires performing linear algebra operations

Great reference:

[optional] Anderson and Moore, Linear Quadratic Methods --- standard reference for LQ setting

- Note: strong similarity with Kalman filtering, which is able to compute the Bayes' filter updates exactly even though in general there are no closed form solutions and numerical solutions scale poorly with dimensionality.

Linear Quadratic Regulator (LQR)

The LQR setting assumes a linear dynamical system:

$$x_{t+1} = Ax_t + Bu_t,$$

x_t: state at time t
u_t: input at time t
It assumes a quadratic cost function:

$$g(x_t, u_t) = x_t^\top Q x_t + u_t^\top R u_t$$

with $Q \succ 0, R \succ 0$.

For a square matrix X we have $X \succ 0$ if and only if for all vectors z we have $z^\top X z > 0$. Hence there is a non-zero cost for any state different from the all-zeros state, and any input different from the all-zeros input.

While LQ assumptions might (at first) seem very restrictive, we will see the method can be made applicable for non-linear systems, e.g., helicopter.

Value Iteration

- Back-up step for i+1 steps to go:

$$J_{i+1}(s) = \min_u g(s,u) + \sum_{s'} P(s'|s,u) J_i(s')$$

- LQR:

$$J_{i+1}(x) = \min_u x^\top Q x + u^\top R u + \sum_{x'=Ax+Bu} J_i(x')$$

$$= \min_u \left[x^\top Q x + u^\top R u + J_i(Ax+Bu) \right]$$

LQR value iteration: J_1

$$J_{i+1}(x) \leftarrow \min_u \left[x^\top Q x + u^\top R u + J_i(Ax + Bu) \right]$$

Initialize $J_0(x) = x^\top P_0 x$.

$$\begin{aligned} J_1(x) &= \min_u \left[x^\top Q x + u^\top R u + J_0(Ax + Bu) \right] \\ &= \min_u \left[x^\top Q x + u^\top R u + (Ax + Bu)^\top P_0 (Ax + Bu) \right] \qquad (1) \end{aligned}$$

To find the minimum over u, we set the gradient w.r.t. u equal to zero:

$$\nabla_u [\ldots] = 2Ru + 2B^\top P_0 (Ax + Bu) = 0,$$

$$\text{hence: } u = -(R + B^\top P_0 B)^{-1} B^\top P_0 A x \qquad (2)$$

$$\begin{aligned} \text{(2) into (1): } J_1(x) &= x^\top P_1 x \\ \text{for: } P_1 &= Q + K_1^\top R K_1 + (A + BK_1)^\top P_0 (A + BK_1) \\ K_1 &= -(R + B^\top P_0 B)^{-1} B^\top P_0 A. \end{aligned}$$

LQR value iteration: J_1 (ctd)

- In summary:

$$\begin{aligned} &J_0(x) = x^\top P_0 x \\ &x_{t+1} = A x_t + B u_t \\ &g(x, u) = u^\top R u + x^\top Q x \end{aligned}$$

$$\begin{aligned} J_1(x) &= x^\top P_1 x \\ \text{for: } P_1 &= Q + K_1^\top R K_1 + (A + BK_1)^\top P_0 (A + BK_1) \\ K_1 &= -(R + B^\top P_0 B)^{-1} B^\top P_0 A. \end{aligned}$$

- $J_1(x)$ is quadratic, just like $J_0(x)$.

  →Value iteration update is the same for all times and can be done in closed form for this particular continuous state-space system and cost!

$$\begin{aligned} J_2(x) &= x^\top P_2 x \\ \text{for: } P_2 &= Q + K_2^\top R K_2 + (A + BK_2)^\top P_1 (A + BK_2) \\ K_2 &= -(R + B^\top P_1 B)^{-1} B^\top P_1 A. \end{aligned}$$

Value iteration solution to LQR

Set $P_0 = 0$.
for $i = 1, 2, 3, \ldots$

$$\begin{aligned} K_i &= -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A \\ P_i &= Q + K_i^\top R K_i + (A + BK_i)^\top P_{i-1} (A + BK_i) \end{aligned}$$

The optimal policy for a i-step horizon is given by:

$$\pi(x) = K_i x$$

The cost-to-go function for a i-step horizon is given by:

$$J_i(x) = x^\top P_i x.$$

LQR assumptions revisited

$$\begin{aligned} x_{t+1} &= A x_t + B u_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

= for keeping a linear system at the all-zeros state while preferring to keep the control input small.

- Extensions which make it more generally applicable:
  - Affine systems
  - System with stochasticity
  - Regulation around non-zero fixed point for non-linear systems
  - Penalization for change in control inputs
  - Linear time varying (LTV) systems
  - Trajectory following for non-linear systems

LQR Ext0: Affine systems

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + c \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

- Optimal control policy remains linear, optimal cost-to-go function remains quadratic
- Two avenues to do derivation:
  - 1. Re-derive the update, which is very similar to what we did for standard setting
  - 2. Re-define the state as: $z_t = [x_t; 1]$, then we have:

$$z_{t+1} = \begin{bmatrix} x_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A & c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t = A' z_t + B' u_t$$

LQR Ext1: stochastic system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \\ & \quad w_t, t = 0, 1, \ldots \text{ are zero mean and independent} \end{aligned}$$

- Exercise: work through similar derivation as we did for the deterministic case.
- Result:
  - Same optimal control policy
  - Cost-to-go function is almost identical: has one additional term which depends on the variance in the noise (and which cannot be influenced by the choice of control inputs)

LQR Ext2: non-linear systems

Nonlinear system: $x_{t+1} = f(x_t, u_t)$

We can keep the system at the state x^* iff

$$\exists u^* \text{s.t.} \quad x^* = f(x^*, u^*)$$

Linearizing the dynamics around x^* gives:

$$x_{t+1} \approx f(x^*, u^*) + \underbrace{\frac{\partial f}{\partial x}(x^*, u^*)}_{A}(x_t - x^*) + \underbrace{\frac{\partial f}{\partial u}(x^*, u^*)}_{B}(u_t - u^*)$$

Equivalently:

$$x_{t+1} - x^* \approx A(x_t - x^*) + B(u_t - u^*)$$

Let $z_t = x_t - x^*$, let $v_t = u_t - u^*$, then:

$$z_{t+1} = Az_t + Bv_t, \qquad \text{cost} = z_t^\top Q z_t + v_t^\top R v_t$$ [=standard LQR]

$$v_t = K z_t \Rightarrow u_t - u^* = K(x_t - x^*) \Rightarrow u_t = u^* + K(x_t - x^*)$$

LQR Ext3: penalize for change in control inputs

- Standard LQR:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

- When run in this format on real systems: often high frequency control inputs get generated. Typically highly undesirable and results in poor control performance.
- Why?
- Solution: frequency shaping of the cost function. Can be done by augmenting the system with a filter and then the filter output can be used in the quadratic cost function. (See, e.g., Anderson and Moore.)
- Simple special case which works well in practice: penalize for change in control inputs. ---- How ??

LQR Ext3: penalize for change in control inputs

- Standard LQR:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

- How to incorporate the change in controls into the cost/ reward function?
  - Soln. method A: explicitly incorporate into the state by augmenting the state with the past control input vector, and the difference between the last two control input vectors.
  - Soln. method B: change of variables to fit into the standard LQR setting.

LQR Ext3: penalize for change in control inputs

- Standard LQR:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

- Introducing change in controls Δu:

$$\underbrace{\begin{bmatrix} x_{t+1} \\ u_{t+1} \end{bmatrix}}_{x'_{t+1}} = \underbrace{\begin{bmatrix} A & B \\ 0 & I \end{bmatrix}}_{A'} \underbrace{\begin{bmatrix} x_t \\ u_{t-1} \end{bmatrix}}_{x'_t} + \underbrace{\begin{bmatrix} B \\ I \end{bmatrix}}_{B'} \underbrace{\Delta u_t}_{u'_t}$$

$$x'_{t+1} = A' \quad x'_t + B' \quad u'_t$$

$$\text{cost} = -(x'^\top Q' x' + \Delta u^\top R' \Delta u) \qquad Q' = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix}$$

R' = penalty for change in controls

[If R'=0, then equivalent to standard LQR.]

LQR Ext4: Linear Time Varying (LTV) Systems

$$
\begin{aligned}
x_{t+1} &= A_t x_t + B_t u_t \\
g(x_t, u_t) &= x_t^\top Q_t x_t + u_t^\top R_t u_t
\end{aligned}
$$

LQR Ext4: Linear Time Varying (LTV) Systems

Set $P_0 = 0$.
for $i = 1, 2, 3, \ldots$

$$
\begin{aligned}
K_i &= -(R_{H-i} + B_{H-i}^\top P_{i-1} B_{H-i})^{-1} B_{H-i}^\top P_{i-1} A_{H-i} \\
P_i &= Q_{H-i} + K_i^\top R_{H-i} K_i + (A_{H-i} + B_{H-i} K_i)^\top P_{i-1} (A_{H-i} + B_{H-i} K_i)
\end{aligned}
$$

The optimal policy for a i-step horizon is given by:

$$\pi(x) = K_i x$$

The cost-to-go function for a i-step horizon is given by:

$$J_i(x) = x^\top P_i x.$$

LQR Ext5: Trajectory following for non-linear systems

- A state sequence $x_0^*, x_1^*, \ldots, x_H^*$ is a feasible target trajectory iff

$$\exists u_0^*, u_1^*, \ldots, u_{H-1}^* \ : \ \forall t \in \{0, 1, \ldots, H-1\} \ : \ x_{t+1}^* = f(x_t^*, u_t^*)$$

- Problem statement:

$$\min_{u_0, u_1, \ldots, u_{H-1}} \sum_{t=0}^{H-1} (x_t - x_t^*)^\top Q (x_t - x_t^*) + (u_t - u_t^*)^\top R (u_t - u_t^*)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

- Transform into linear time varying case (LTV):

$$x_{t+1} \approx f(x_t^*, u_t^*) + \underbrace{\frac{\partial f}{\partial x}(x_t^*, u_t^*)}_{A_t}(x_t - x_t^*) + \underbrace{\frac{\partial f}{\partial u}(x_t^*, u_t^*)}_{B_t}(u_t - u_t^*)$$

$$x_{t+1} - x_{t+1}^* \approx A_t (x_t - x_t^*) + B_t (u_t - u_t^*)$$

LQR Ext5: Trajectory following for non-linear systems

- Transformed into linear time varying case (LTV):

$$\min_{u_0, u_1, \ldots, u_{H-1}} \sum_{t=0}^{H-1} (x_t - x_t^*)^\top Q (x_t - x_t^*) + (u_t - u_t^*)^\top R (u_t - u_t^*)$$

$$\text{s.t.} \ x_{t+1} - x_{t+1}^* = A_t (x_t - x_t^*) + B_t (u_t - u_t^*)$$

- Now we can run the standard LQR back-up iterations.
- Resulting policy at i time-steps from the end:

$$u_{H-i} - u_{H-i}^* = K_i (x_{H-i} - x_{H-i}^*)$$

- The target trajectory need not be feasible to apply this technique, however, if it is infeasible then the linearizations are not around the (state,input) pairs that will be visited

Most general cases

- Methods which attempt to solve the generic optimal control problem

$$\begin{array}{ll} \min_u & \sum_{t=0}^{H} g(x_t, u_t) \\ \text{subject to} & x_{t+1} = f(x_t, u_t) \ \ \forall t \end{array}$$

by iteratively approximating it and leveraging the fact that the linear quadratic formulation is easy to solve.

Iteratively apply LQR

Initialize the algorithm by picking either (a) A control policy $\pi^{(0)}$ or (b) A sequence of states $x_0^{(0)}, x_1^{(0)}, \ldots, x_H^{(0)}$ and control inputs $u_0^{(0)}, u_1^{(0)}, \ldots, u_H^{(0)}$. With initialization (a), start in Step (1). With initialization (b), start in Step (2).

Iterate the following:

(1) Execute the current policy $\pi^{(i)}$ and record the resulting state-input trajectory $x_0^{(i)}, u_0^{(i)}, x_1^{(i)}, u_1^{(i)}, \ldots, x_H^{(i)}, u_H^{(i)}$.

(2) Compute the LQ approximation of the optimal control problem around the obtained state-input trajectory by computing a first-order Taylor expansion of the dynamics model, and a second-order Taylor expansion of the cost function.

(3) Use the LQR back-ups to solve for the optimal control policy $\pi^{(i+1)}$ for the LQ approximation obtained in Step (2).

(4) Set $i = i + 1$ and go to Step (1).

Iterative LQR: in standard LTV format

Standard LTV is of the form $z_{t+1} = A_t z_t + B_t v_t$, $g(z, v) = z^\top Q z + v^\top R v$.

Linearizing around $(x_t^{(i)}, u_t^{(i)})$ in iteration i of the iterative LQR algorithm gives us (up to first order!):

$$x_{t+1} = f(x_t^{(i)}, u_t^{(i)}) + \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)})(x_t - x_t^{(i)}) + \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)})(u_t - u_t^{(i)})$$

Subtracting the same term on both sides gives the format we want:

$$x_{t+1} - x_{t+1}^{(i)} = f(x_t^{(i)}, u_t^{(i)}) - x_{t+1}^{(i)} + \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)})(x_t - x_t^{(i)}) + \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)})(u_t - u_t^{(i)})$$

Hence we get the standard format if using:

$$\begin{aligned}
z_t &= [x_t - x_t^{(i)} \quad 1]^\top \\
v_t &= (u_t - u_t^{(i)}) \\
A_t &= \begin{bmatrix} \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)}) & f(x_t^{(i)}, u_t^{(i)}) - x_{t+1}^{(i)} \\ 0 & 1 \end{bmatrix} \\
B_t &= \begin{bmatrix} \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)}) \\ 0 \end{bmatrix}
\end{aligned}$$

A similar derivation is needed to find Q and R.

Iteratively apply LQR: convergence

- Need not converge as formulated!
  - Reason: the optimal policy for the LQ approximation might end up not staying close to the sequence of points around which the LQ approximation was computed by Taylor expansion.
  - Solution: in each iteration, adjust the cost function so this is the case, i.e., use the cost function

    $$(1-\alpha)g(x_t, u_t) + \alpha(\|x_t - x_t^{(i)}\|_2^2 + \|u_t - u_t^{(i)}\|_2^2)$$

    Assuming g is bounded, for α close enough to one, the 2nd term will dominate and ensure the linearizations are good approximations around the solution trajectory found by LQR.

Iteratively apply LQR: practicalities

- f is non-linear, hence this is a non-convex optimization problem. Can get stuck in local optima! Good initialization matters.
- g could be non-convex: Then the LQ approximation fails to have positive-definite cost matrices.
  - Practical fix: if Q_t or R_t are not positive definite → increase penalty for deviating from current state and input $(x^{(i)}_t, u^{(i)}_t)$ until resulting Q_t and R_t are positive definite.

Iterative LQR for trajectory following

While there is no need to follow this particular route, this is a (imho) particularly convenient way of turning the linearized and quadraticized approximation in the iLQR iterations into the standard LQR format for the setting of trajectory following with a quadratic penalty for deviation from the trajectory.

Let $x_t^{(i)}, u_t^{(i)}$ be the state and control around which we linearize. Let x_t^*, u_t^* be the target controls then we have:

$$\begin{aligned}
x_{t+1} &= f(x_t^{(i)}, u_t^{(i)}) + \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)})(x_t - x_t^{(i)}) + \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)})(u_t - u_t^{(i)}) \\
x_{t+1} - x_{t+1}^* &= f(x_t^{(i)}, u_t^{(i)}) - x_{t+1}^* + \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)})(x_t - x_t^{(i)} - x_t^* + x_t^*) + \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)})(u_t - u_t^{(i)} - u_t^* + u_t^*) \\
x_{t+1} - x_{t+1}^* &= f(x_t^{(i)}, u_t^{(i)}) - x_{t+1}^* + \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)})(x_t - x_t^*) + \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)})(x_t^* - x_t^{(i)}) \\
&\quad + \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)})(u_t - u_t^*) + \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)})(u_t^* - u_t^{(i)}) \\
[x_{t+1} - x_{t+1}^*; 1] &= A[(x_t - x_t^*); 1] + B(u_t - u_t^*)
\end{aligned}$$

For

$$A = \begin{bmatrix} \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)}) & f(x_t^{(i)}, u_t^{(i)}) - x_{t+1}^* + \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)})(x_t^* - x_t^{(i)}) + \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)})(u_t^* - u_t^{(i)}) \\ 0 & 1 \end{bmatrix}$$

and

$$B = \begin{bmatrix} \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)}) \\ 0 \end{bmatrix}$$

The cost function can be used as is: $(x_t - x_t^*)^\top Q(x_t - x_t^*) + (u_t - u_t^*)^\top R(u_t - u_t^*)$.

Differential Dynamic Programming (DDP)

- Often loosely used to refer to iterative LQR procedure.
- More precisely: Directly perform 2nd order Taylor expansion of the Bellman back-up equation [rather than linearizing the dynamics and 2nd order approximating the cost]
- Turns out this retains a term in the back-up equation which is discarded in the iterative LQR approach
- [It's a quadratic term in the dynamics model though, so even if cost is convex, resulting LQ problem could be non-convex …]

[Reference: Jacobson and Mayne, "Differential dynamic programming," 1970]

Differential dynamic programming

$$
\begin{aligned}
J_{i+1}(x) \quad = \quad & \min_u \\
& \text{2nd order expansion of } g \text{ around } (x^*, u^*) \\
& + J_i(f(x^*, u^*)) \\
& + \frac{dJ}{dx}(f(x,u) - f(x^*, u^*)) \\
& + (f(x,u) - f(x^*, u^*))^\top \frac{d^2 J}{dx^2}(f(x,u) - f(x^*, u^*))
\end{aligned}
$$

To keep entire expression 2nd order:
Use Taylor expansions of f and then remove all resulting terms which are higher than 2nd order.
Turns out this keeps 1 additional term compared to iterative LQR

Can we do even better?

- Yes!
- At convergence of iLQR and DDP, we end up with linearizations around the (state,input) trajectory the algorithm converged to
- In practice: the system could not be on this trajectory due to perturbations / initial state being off / dynamics model being off / …
- Solution: at time t when asked to generate control input ut, we could re-solve the control problem using iLQR or DDP over the time steps t through H
- Replanning entire trajectory is often impractical → in practice: replan over horizon h. = ***receding horizon control***
  - This requires providing a cost to go $J^{(t+h)}$ which accounts for all future costs. This could be taken from the offline iLQR or DDP run

Multiplicative noise

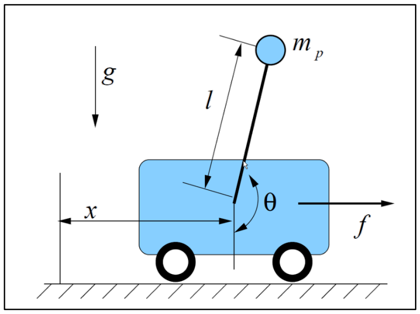
- In many systems of interest, there is noise entering the system which is multiplicative in the control inputs, i.e.:

$$x_{t+1} = Ax_t + (B + B_w w_t)u_t$$

- Exercise: LQR derivation for this setting

[optional related reading:Todorov and Jordan, nips 2003]

Cart-pole

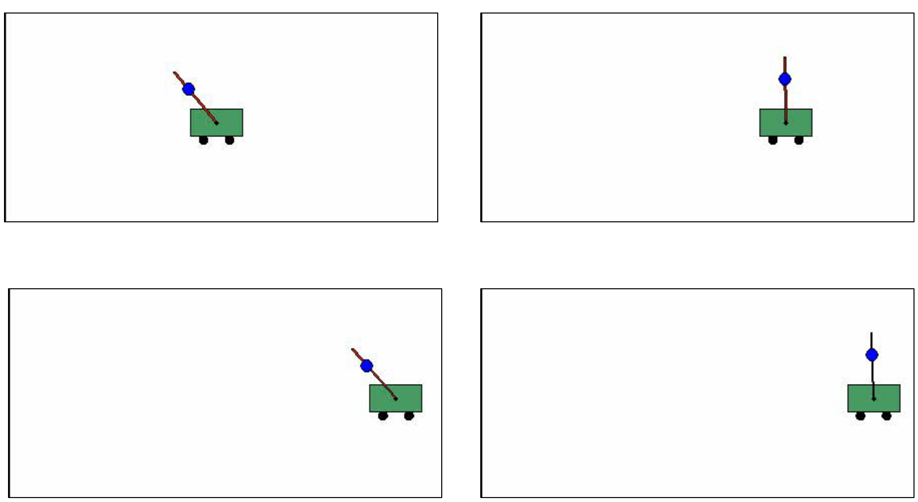


$$\boxed{H(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)u}$$

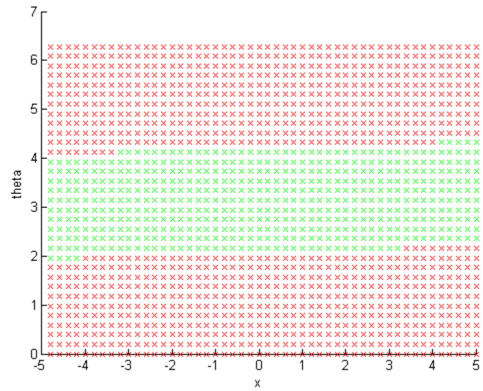
$$
\begin{aligned}
H(q) &= \begin{bmatrix} m_c + m_p & m_p l \cos\theta \\ m_p l \cos\theta & m_p l^2 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} 0 & -m_p l \dot{\theta} \sin\theta \\ 0 & 0 \end{bmatrix} \\
G(q) &= \begin{bmatrix} 0 \\ m_p g l \sin\theta \end{bmatrix} \\
B &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\end{aligned}
$$

[See also Section 3.3 in Tedrake notes.]

Cart-pole --- LQR

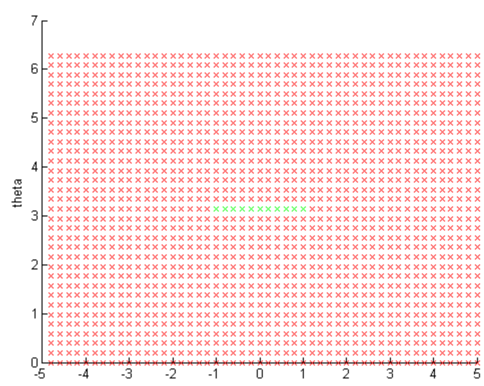


Cart-pole --- LQR



Q = diag([1;1;1;1]); R = 0; [x, theta, xdot, thetadot]

Cart-pole --- LQR



Q = diag([1;1;1;1]); R = 1; [x, theta, xdot, thetadot]

[See, e.g., Slotine and Li, or Boyd lecture notes (pointers available on course website) if you want to find out more.]

Lyapunov's linearization method

Defn. x^* is an asymptotically stable equilibrium point for system f if there exists an $\epsilon > 0$ such that for all initial states x s.t. $\| x - x^* \| \leq \epsilon$ we have that $\lim_{t \to \infty} x_t = x^*$

We will not cover any details, but here is the basic result:

Assume x^* is an equilibrium point for f(x), i.e., $x^* = f(x^*)$.

If x^* is an asymptotically stable equilibrium point for the linearized system, then it is asymptotically stable for the non-linear system.

If x^* is unstable for the linear system, it's unstable for the non-linear system.

If x^* is marginally stable for the linear system, no conclusion can be drawn.

= additional justification for linear control design techniques

Controllability

- A system is t-time-steps controllable if from any start state, x_0, we can reach any target state, x^*, at time t.
- For a linear time-invariant systems, we have:

$$x_t = A^t x_0 + A^{t-1} B u_0 + A^{t-2} B u_1 + \ldots + AB u_{t-2} + B u_{t-1}$$

hence the system is t-time-steps controllable if and only if the above linear system of equations in $u_0, \ldots, u_{t-1}$ has a solution for all choices of x_0 and x_t. This is the case if and only if

$$\text{rank} \begin{bmatrix} A^{t-1}B & A^{t-2}B & \cdots & A^2B & AB & B \end{bmatrix} = n$$

with n the dimension of the statespace.

The Cayley-Hamilton theorem from linear algebra says that for all A, for all $t \geq n$:

$$\exists w \in \mathbb{R}^n, \quad A^t = \sum_{i=0}^{n-1} w_i A^i$$

Hence we obtain that the system (A,B) is controllable for all times t>=n, if and only if

$$\text{rank} \begin{bmatrix} A^{n-1}B & A^{n-2}B & \cdots & A^2B & AB & B \end{bmatrix} = n$$

Feedback linearization

Consider system of the form:

$$\dot{x} = f(x) + g(x)u$$

If $g(x)$ is square (i.e., number of control inputs = number of state variables) and it is invertible, then we can linearize the system by a change of input variables:

$$v = f(x) + g(x)u$$

gives us:

$$\dot{x} = v$$

Prototypical example: fully actuated manipulators:

$$H(q)\ddot{q} + b(q, \dot{q}) + g(q) = \tau$$

Feedback linearize by using the following transformed input:

$$v = H^{-1}(q)\left(\tau - g(q) - b(q, \dot{q})\right)$$

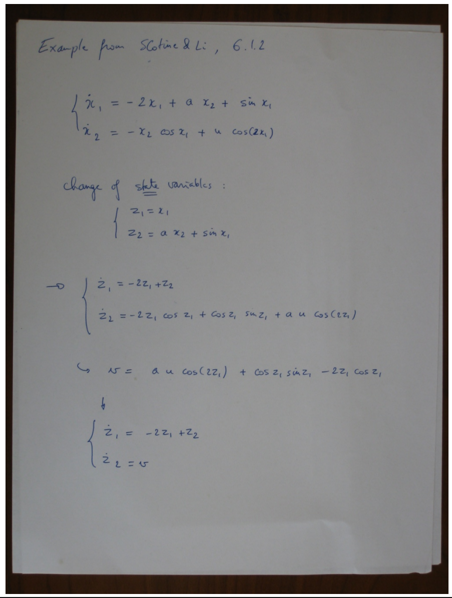
which results in

$$\ddot{q} = v$$

Feedback linearization

$$\begin{aligned}
\dot{x}_1 &= -2x_1 + ax_2 + \sin x_1 \\
\dot{x}_2 &= -x_2 \cos x_1 + u \cos(2x_1)
\end{aligned}$$

Feedback linearization



Example from Slotine & Li, 6.1.2

$$\begin{cases} \dot{x}_1 = -2x_1 + a\,x_2 + \sin x_1 \\ \dot{x}_2 = -x_2 \cos x_1 + u \cos(2x_1) \end{cases}$$

change of state variables :

$$\begin{cases} z_1 = x_1 \\ z_2 = a\,x_2 + \sin x_1 \end{cases}$$

$$\rightarrow \begin{cases} \dot{z}_1 = -2z_1 + z_2 \\ \dot{z}_2 = -2z_1 \cos z_1 + \cos z_1 \sin z_1 + a\,u \cos(2z_1) \end{cases}$$

$$\hookrightarrow \quad v = a\,u \cos(2z_1) + \cos z_1 \sin z_1 - 2z_1 \cos z_1$$

$$\downarrow$$

$$\begin{cases} \dot{z}_1 = -2z_1 + z_2 \\ \dot{z}_2 = v \end{cases}$$

Feedback linearization

$\dot{x} = f(x) + g(x)u$ (6.52)

Definition 6.6 *A single-input nonlinear system in the form (6.52), with* $\mathbf{f(x)}$ *and* $\mathbf{g(x)}$ *being smooth vector fields on* $\mathbf{R}^n$*, is said to be* *input-state linearizable* *if there exists a region* Ω *in* $\mathbf{R}^n$*, a diffeomorphism* $\phi : \Omega \rightarrow \mathbf{R}^n$*, and a nonlinear feedback control law*

$$u = \alpha(\mathbf{x}) + \beta(\mathbf{x})v \qquad (6.53)$$

such that the new state variables $\mathbf{z} = \phi(\mathbf{x})$ *and the new input* v *satisfy a linear time-invariant relation*

$$\dot{\mathbf{z}} = \mathbf{A}\,\mathbf{z} + \mathbf{b}\,v \qquad (6.54)$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & . & . & 0 \\ 0 & 0 & 1 & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 1 \\ 0 & 0 & 0 & . & . & 0 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 1 \end{bmatrix}$$

[A function is called a diffeomorphism if it is smooth and its inverse is smooth.]

[From: Slotine and Li]

Feedback linearization

Theorem 6.2 *The nonlinear system (6.52), with* $\mathbf{f}(\mathbf{x})$ *and* $\mathbf{g}(\mathbf{x})$ *being smooth vector fields, is input-state linearizable if, and only if, there exists a region* $\boldsymbol{\Omega}$ *such that the following conditions hold:*

- *the vector fields* $\{\mathbf{g}, ad_{\mathbf{f}}\,\mathbf{g}, \ldots, ad_{\mathbf{f}}^{\,n-1}\,\mathbf{g}\}$ *are linearly independent in* $\boldsymbol{\Omega}$
- *the set* $\{\mathbf{g}, ad_{\mathbf{f}}\,\mathbf{g}, \ldots, ad_{\mathbf{f}}^{\,n-2}\,\mathbf{g}\}$ *is involutive in* $\boldsymbol{\Omega}$

Definition 6.1 *Let* $h : \mathbf{R}^n \to \mathbf{R}$ *be a smooth scalar function, and* $\mathbf{f} : \mathbf{R}^n \to \mathbf{R}^n$ *be a smooth vector field on* $\mathbf{R}^n$, *then the Lie derivative of h with respect to* $\mathbf{f}$ *is a scalar function defined by* $L_{\mathbf{f}}\, h = \nabla h\ \mathbf{f}$.

Thus, the Lie derivative $L_{\mathbf{f}}\, h$ is simply the directional derivative of h along the direction of the vector $\mathbf{f}$.

Repeated Lie derivatives can be defined recursively

$$L_{\mathbf{f}}^{\,o}\, h = h$$

$$L_{\mathbf{f}}^{\,i}\, h = L_{\mathbf{f}}(L_{\mathbf{f}}^{\,i-1}\, h) = \nabla(L_{\mathbf{f}}^{\,i-1}\, h)\ \mathbf{f} \qquad \text{for } i = 1, 2, \ldots..$$

Similarly, if $\mathbf{g}$ is another vector field, then the scalar function $L_{\mathbf{g}}\, L_{\mathbf{f}}\, h(\mathbf{x})$ is

$$L_{\mathbf{g}}\, L_{\mathbf{f}}\, h = \nabla(L_{\mathbf{f}}\, h)\ \mathbf{g}$$

Feedback linearization

Theorem 6.2 *The nonlinear system (6.52), with* $\mathbf{f}(\mathbf{x})$ *and* $\mathbf{g}(\mathbf{x})$ *being smooth vector fields, is input-state linearizable if, and only if, there exists a region* $\boldsymbol{\Omega}$ *such that the following conditions hold:*

- *the vector fields* $\{\mathbf{g}, ad_{\mathbf{f}}\,\mathbf{g}, \ldots, ad_{\mathbf{f}}^{\,n-1}\,\mathbf{g}\}$ *are linearly independent in* $\boldsymbol{\Omega}$
- *the set* $\{\mathbf{g}, ad_{\mathbf{f}}\,\mathbf{g}, \ldots, ad_{\mathbf{f}}^{\,n-2}\,\mathbf{g}\}$ *is involutive in* $\boldsymbol{\Omega}$

Definition 6.2 *Let* $\mathbf{f}$ *and* $\mathbf{g}$ *be two vector fields on* $\mathbf{R}^n$. *The Lie bracket of* $\mathbf{f}$ *and* $\mathbf{g}$ *is a third vector field defined by*

$$[\mathbf{f}, \mathbf{g}] = \nabla\mathbf{g}\ \mathbf{f} - \nabla\mathbf{f}\ \mathbf{g}$$

The Lie bracket $[\mathbf{f}, \mathbf{g}]$ is commonly written as $ad_{\mathbf{f}}\,\mathbf{g}$ (where *ad* stands for "adjoint"). Repeated Lie brackets can then be defined recursively by

$$ad_{\mathbf{f}}^{\,o}\,\mathbf{g} = \mathbf{g}$$

$$ad_{\mathbf{f}}^{\,i}\,\mathbf{g} = [\mathbf{f}, ad_{\mathbf{f}}^{\,i-1}\,\mathbf{g}] \qquad \text{for } i = 1, 2, \ldots..$$

Feedback linearization

Theorem 6.2 *The nonlinear system (6.52), with* $\mathbf{f}(\mathbf{x})$ *and* $\mathbf{g}(\mathbf{x})$ *being smooth vector fields, is input-state linearizable if, and only if, there exists a region* Ω *such that the following conditions hold:*

- *the vector fields* $\{\mathbf{g}, ad_{\mathbf{f}}\,\mathbf{g}, \ldots, ad_{\mathbf{f}}^{\,n-1}\,\mathbf{g}\}$ *are linearly independent in* Ω
- *the set* $\{\mathbf{g}, ad_{\mathbf{f}}\,\mathbf{g}, \ldots, ad_{\mathbf{f}}^{\,n-2}\,\mathbf{g}\}$ *is involutive in* Ω

Definition 6.5 *A linearly independent set of vector fields* $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m\}$ *is said to be involutive if, and only if, there are scalar functions* $\alpha_{ijk} : \mathbf{R}^n \to \mathbf{R}$ *such that*

$$[\mathbf{f}_i, \mathbf{f}_j](\mathbf{x}) = \sum_{k=1}^{m} \alpha_{ijk}(\mathbf{x})\, \mathbf{f}_k(\mathbf{x}) \qquad \forall\, i, j \tag{6.51}$$

Feedback linearization

Theorem 6.2 *The nonlinear system (6.52), with* $\mathbf{f}(\mathbf{x})$ *and* $\mathbf{g}(\mathbf{x})$ *being smooth vector fields, is input-state linearizable if, and only if, there exists a region* Ω *such that the following conditions hold:*

- *the vector fields* $\{\mathbf{g}, ad_{\mathbf{f}}\,\mathbf{g}, \ldots, ad_{\mathbf{f}}^{\,n-1}\,\mathbf{g}\}$ *are linearly independent in* Ω
- *the set* $\{\mathbf{g}, ad_{\mathbf{f}}\,\mathbf{g}, \ldots, ad_{\mathbf{f}}^{\,n-2}\,\mathbf{g}\}$ *is involutive in* Ω

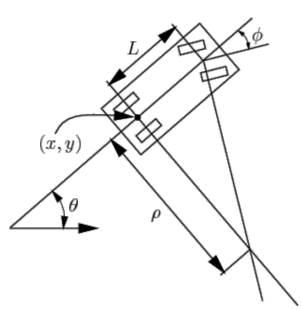
→This condition can be checked by applying the chain rule and examining the rank of certain matrices!

→ The proof is actually semi-constructive: it constructs a set of partial differential equations to which the state transformation is the solution.

Feedback linearization

- Further readings:
  - Slotine and Li, Chapter 6 – example 6.10 shows state-input linearization in action
  - Isidori, Nonlinear control systems, 1989.

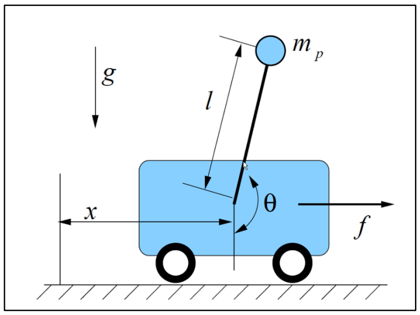
Car



$$\dot{x} = u_s \cos\theta$$
$$\dot{y} = u_s \sin\theta$$
$$\dot{\theta} = \frac{u_s}{L} \tan u_\phi.$$

- For your reference: Standard (kinematic) car models: (From Lavalle, Planning Algorithms, 2006, Chapter 13)
  - Tricycle: $u_S \in [-1, 1], u_\phi \in [-\pi/2, \pi/2]$
  - Simple Car: $u_S \in [-1, 1], u_\phi \in [-\phi_{\max}, \phi_{\max}], \phi_{\max} < \pi/2$
  - Reeds-Shepp Car: $u_S \in \{-1, 0, 1\}, u_\phi \in [-\phi_{\max}, \phi_{\max}], \phi_{\max} < \pi/2$
  - Dubins Car: $u_S \in \{0, 1\}, u_\phi \in [-\phi_{\max}, \phi_{\max}], \phi_{\max} < \pi/2$

Cart-pole

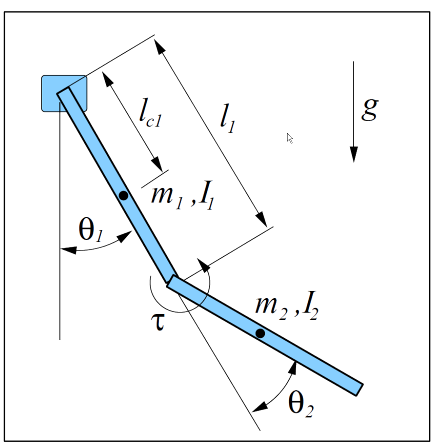


$$\boxed{H(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)u}$$

$$\begin{aligned}
H(q) &= \begin{bmatrix} m_c + m_p & m_p l \cos\theta \\ m_p l \cos\theta & m_p l^2 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} 0 & -m_p l \dot{\theta} \sin\theta \\ 0 & 0 \end{bmatrix} \\
G(q) &= \begin{bmatrix} 0 \\ m_p g l \sin\theta \end{bmatrix} \\
B &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\end{aligned}$$

[See also Section 3.3 in Tedrake notes.]

Acrobot



$$\boxed{H(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)u}$$

$$\begin{aligned}
H(q) &= \begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + 2 m_2 l_1 l_{c_2} c_2 & I_2 + m_2 L_1 l_{c2} c_2 \\ I_2 + m_2 l_1 l_{c2} c_2 & I_2 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} -2 m_2 l_1 l_{c2} s_2 \dot{q}_2 & -m_2 l_1 l_{c2} s_2 \dot{q}_2 \\ m_2 l_1 l_{c2} s_2 \dot{q}_1 & 0 \end{bmatrix} \\
G(q) &= \begin{bmatrix} (m_1 l_{c1} + m_2 l_1) g s_1 + m_2 g l_2 s_{1+2} \\ m_2 g l_2 s_{1+2} \end{bmatrix} \\
B &= \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\end{aligned}$$

[See also Section 3.2 in Tedrake notes.]

Lagrangian dynamics

- Newton: F = ma
  - Quite generally applicable
  - Its application can become a bit cumbersome in multi-body systems with constraints/internal forces
- Lagrangian dynamics method eliminates the internal forces from the outset and expresses dynamics w.r.t. the degrees of freedom of the system

Lagrangian dynamics

- r_i: generalized coordinates
- T: total kinetic energy
- U: total potential energy
- Q_i : generalized forces $\quad Q_i = \sum_j F_j \frac{dr_i}{dq_j}$
- Lagrangian L = T – U

→ Lagrangian dynamic equations:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i$$

[Nice reference: Goldstein, Poole and Satko, "Classical Mechanics"]

Lagrangian dynamics: point mass example

Consider a point mass m with coordinates (x, y, z) close to earth and with external forces F_x, F_y, F_z.

$$T = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2 + \dot{z}^2)$$

$$U = mgz$$

Lagrangian dynamic equations:

$$\begin{aligned}
F_x &= \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = m\ddot{x} \\
F_y &= \frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = m\ddot{y} \\
F_z &= \frac{d}{dt}\frac{\partial L}{\partial \dot{z}} - \frac{\partial L}{\partial z} = m\ddot{z} - mg
\end{aligned}$$

Lagrangian dynamics: simple double pendulum

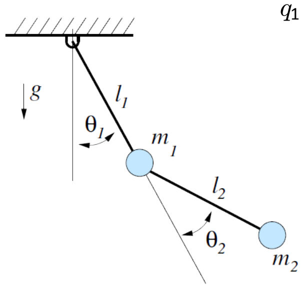


FIGURE A.1 Simple Double Pendulum

$$q_1 = \theta_1, q_2 = \theta_2, s_i = \sin\theta_i, c_i = \cos\theta_i, s_{1+2} = \sin(\theta_1 + \theta_2)$$

$$\begin{aligned}
\mathbf{x}_1 &= \begin{bmatrix} l_1 s_1 \\ -l_1 c_1 \end{bmatrix}, \quad \mathbf{x}_2 = \mathbf{x}_1 + \begin{bmatrix} l_2 s_{1+2} \\ -l_2 c_{1+2} \end{bmatrix} \\
\dot{\mathbf{x}}_1 &= \begin{bmatrix} l_1 \dot{q}_1 c_1 \\ l_1 \dot{q}_1 s_1 \end{bmatrix}, \quad \dot{\mathbf{x}}_2 = \dot{\mathbf{x}}_1 + \begin{bmatrix} l_2(\dot{q}_1 + \dot{q}_2) c_{1+2} \\ l_2(\dot{q}_1 + \dot{q}_2) s_{1+2} \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
T &= \frac{1}{2}\dot{\mathbf{x}}_1^T m_1 \dot{\mathbf{x}}_1 + \frac{1}{2}\dot{\mathbf{x}}_2^T m_2 \dot{\mathbf{x}}_2 \\
&= \frac{1}{2}(m_1 + m_2) l_1^2 \dot{q}_1^2 + \frac{1}{2} m_2 l_2^2 (\dot{q}_1 + \dot{q}_2)^2 + m_2 l_1 l_2 \dot{q}_1 (\dot{q}_1 + \dot{q}_2) c_2 \\
U &= m_1 g y_1 + m_2 g y_2 = -(m_1 + m_2) g l_1 c_1 - m_2 g l_2 c_{1+2}
\end{aligned}$$

$$\begin{aligned}
(m_1 + m_2) l_1^2 \ddot{q}_1 + m_2 l_2^2 (\ddot{q}_1 + \ddot{q}_2) + m_2 l_1 l_2 (2\ddot{q}_1 + \ddot{q}_2) c_2 \\
- m_2 l_1 l_2 (2\dot{q}_1 + \dot{q}_2)\dot{q}_2 s_2 + (m_1 + m_2) l_1 g s_1 + m_2 g l_2 s_{1+2} = \tau_1 \\
m_2 l_2^2 (\ddot{q}_1 + \ddot{q}_2) + m_2 l_1 l_2 \ddot{q}_1 c_2 + m_2 l_1 l_2 \dot{q}_1^2 s_2 + m_2 g l_2 s_{1+2} = \tau_2
\end{aligned}$$

[From: Tedrake Appendix A]