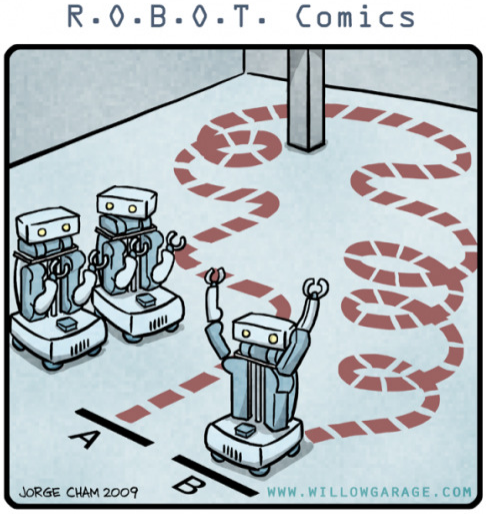
CS 287: Advanced Robotics
Fall 2009

Lecture 6: Control 5: Optimal control --- [Function approximation in dynamic programming---special case: quadratic]

Pieter Abbeel
UC Berkeley EECS

PHD



"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

Announcements

- Will there be lecture this Thursday (Sept 24)?
  - Yes.
- No office hours this Thursday (as I am examining students for prelims).
- Feel free to schedule an appointment by email instead.

Announcements

- Final project contents:
  - Original investigation into an area that relates sufficiently closely to the course.
    - Could be algorithmic/theoretical idea
    - Could be application of existing algorithm(s) to a platform or domain in which these algorithms carry promise but have not been applied
    - Alternatively: Significant improvement for an existing (or new) assignment for this course or for an existing (or new) assignment for 188 which has close ties to this course.
  - Ideally: we are able to identify a topic that relates both to your on-going PhD research and the course.
  - You are very welcome to come up with your own project ideas, yet make sure to pass them by me **before** you submit your abstract.
  - Feel free to stop by office hours or set an appointment (via email) to discuss potential projects.

Announcements

- Final project logistics:
  - Final result: 6-8 page paper.
    - Should be structured like a conference paper, i.e., focus on the problem setting, why it matters, what is interesting/unsolved about it, your approach, results, analysis, and so forth. Cite and briefly survey prior work as appropriate, but don't re-write prior work when not directly relevant to understand your approach.
  - Milestones:
    - Oct. 9th, 23:59: **Approved-by-me** abstracts due: 1 page description of project + goals for milestone. Make sure to sync up with me before then!
    - Nov 9th, 23:59: 1 page milestone report due
    - Dec 3rd, In-class project presentations [tentatively]
    - Dec 11th, 23:59: Final paper due
  - 1 or 2 students/project. If you are two students on 1 final project, I will expect twice as much research effort has gone into it!

Bellman's curse of dimensionality

- n-dimensional state space
- Number of states grows exponentially in n
- In practice
  - Discretization is considered only computationally feasible up to 5 or 6 dimensional state spaces even when using
    - Variable resolution discretization
    - Very fast implementations

Today

- Linear Quadratic (LQ) setting --- special case: can solve continuous optimal control problem exactly

Great reference:

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

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 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:

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

- LQR:

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

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

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

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

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

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

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

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

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!

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

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

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

= for keeping a linear system at the all-zeros state.

- 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. Work through the DP update as we did for standard setting
  - 2. Redefine 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:

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

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

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. (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 and the reward function, and re-do the derivation based upon value iteration.
  - 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:

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

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

$R' =$ 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^*, …, 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):

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

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

LQR Ext5: Trajectory following for non-linear systems

- Transformed into linear time varying case (LTV):

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

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

- 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

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

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 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:

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

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.

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.

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 …]

[Typically cited book: 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

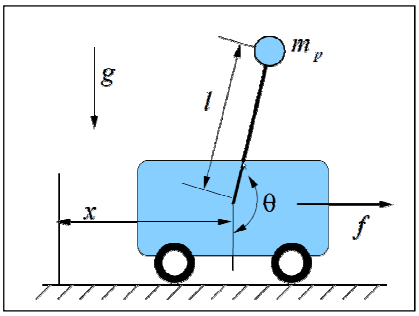
- In many systems of interest, there is noise entering the system which is multiplicative in the control inputs, i.e.:

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

- Exercise: LQR derivation for this setting

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

Cart-pole

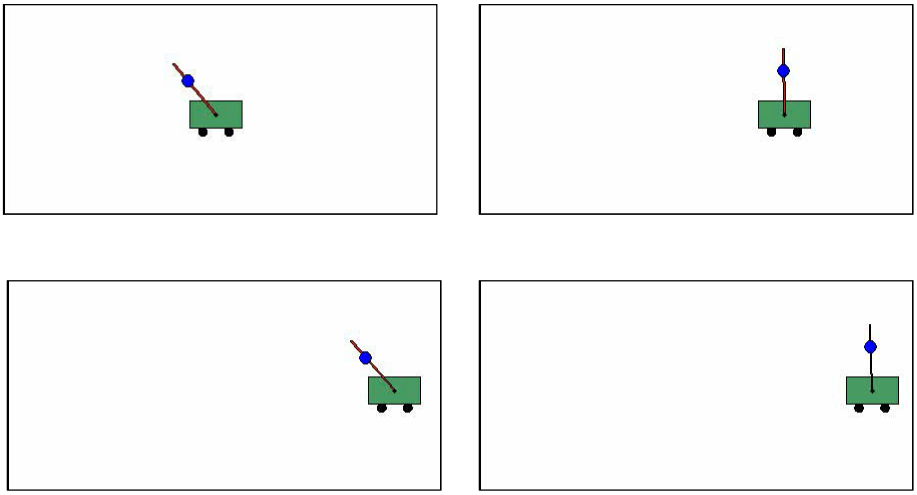


$$H(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)u$$

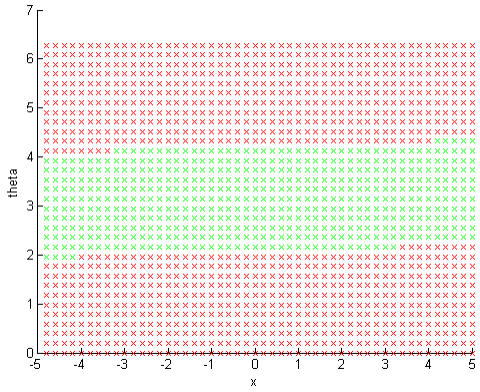
$$\begin{aligned} H(q) &= \begin{bmatrix} m_c + m_p & m_p l\cos\theta \\ m_p l\cos\theta & m_p l^2 \end{bmatrix} \\ C(q,\dot{q}) &= \begin{bmatrix} 0 & -m_p l\dot{\theta}\sin\theta \\ 0 & 0 \end{bmatrix} \\ G(q) &= \begin{bmatrix} 0 \\ m_p g l\sin\theta \end{bmatrix} \\ B &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}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

[See also Section 3.3 in Tedrake notes.]

Cart-pole --- LQR

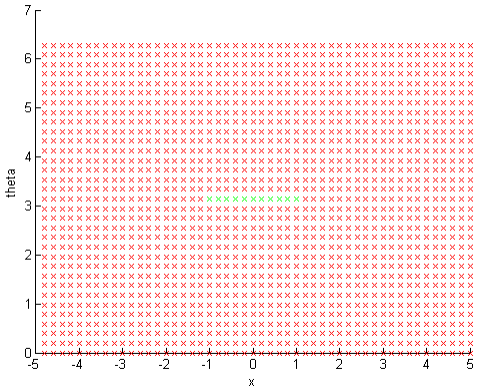


Cart-pole --- LQR



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]

Lyapunov's linearization method

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.*

This provides additional justification for using linear control design techniques for non-linear systems.

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