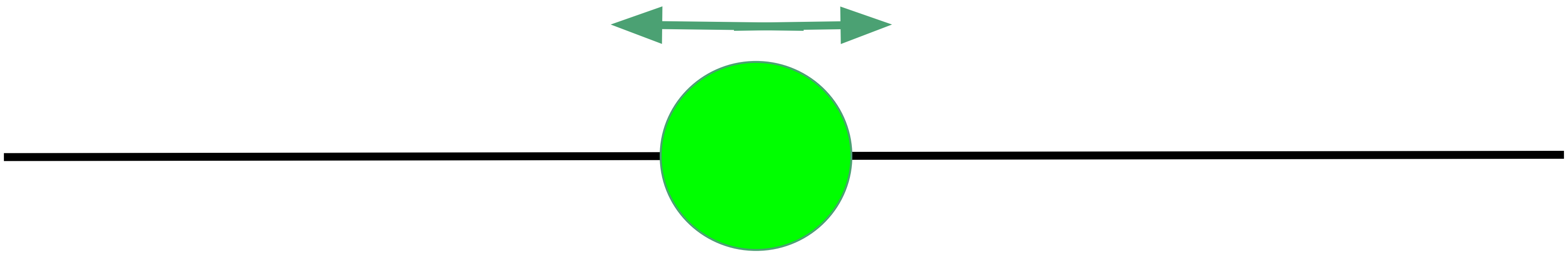
Underactuated Robotics

First, some definitions

- Actuators
  - A single component of a machine that takes a control signal and provides movement.
  - Pneumatics
  - Motors
  - Hydraulics

Degrees of Freedom (DOF)

- Number of parameters that determine the state of a system
- If an object or machine can only move linearly on 1 axis (cannot rotate), then it's position only needs 1 variable to be described. Hence it has 1 DOF.



Fully Actuated VS. Underactuated Machines

- Fully Actuated
  - An actuator for every joint in the machine
  - At Least as many actuators as DOF
- Underactuated
  - Number of Actuators < Number of DOF

Pros VS. Cons

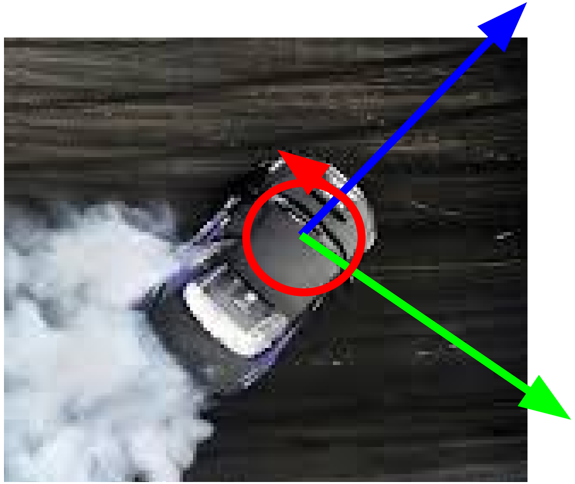
Fully Actuated	Under Actuated
• Simpler physical dynamics • Direct control over every joint and axis • Often results in rigid constrained movement	• More power efficient as it utilizes less actuators • More natural and fluid motion • Requires understanding and research of the natural dynamics of the system • More computationally intensive

Fully Actuated Machine Example (Train)

- Single actuator (engine car)
- Single DOF
- Fully actuated
- Completely rigid motion

Underactuated Machine Example (Car)

- 2 actuators (car engine + steering)
- 3 DOF
- Underactuated
- Longitudinal
- Lateral
- Yaw



Source: http://www.speedhunters.com

Underactuated Robots

- Current robots move too conservatively and only achieve a fraction of their mechanically capable potential
- Limited by control technology
- Focus on building control systems that utilize the natural dynamics of the machine to achieve speed, efficiency, and robustness
- Often requires utilizing feedback in control techniques

Underactuated Robot Example 1

- Control of double pendulum with single actuator
- Use natural dynamics to balance the system with only a single actuator
- Analogy -

Underactuated Robot Example 2

- Single actuator drone (1 propeller)
- Utilize a single rotor, natural dynamics, and feedback to create flight capable drone

Optimal Control via Dynamic Programming

- Define a goal of control as the long-term optimization of a scalar cost function.
- In pendulum case, the goal is to balance the position or reduce its velocity at the maximum height.
- Model the states of the machine as a graph and use dynamic programming to find the best next action.
- [EXPLAIN dynamic programming and in this case how using solutions to different states to solve other states]

Solving the dynamic problem

- Model the robot as a state machine, where each state has an associative cost and actions to move from state to state also incur cost.
- Use value iteration to find the optimal ‘cost-to-go’ to find the minimum cost from all states to the goal state.

Value Iteration

- Let g(s,a) denote the cost of being in state s and taking action a
- Let f(s,a) denote the state transition function so if in state s1, and taking action a1 leads to state a2, then f(s1,a1) = s2
- Then find the optimal cost-to-goal from every node via value iteration DP:

$$\hat{J}^*(s_i) \Leftarrow \min_{a \in A} \left[g(s_i, a) + \hat{J}^*\left(f(s_i, a)\right) \right]$$

- Once we have J*, we can calculate the optimal policy, or the best action to take given a state(s). *a* = *π*∗(*s*)

$$\pi^*(s_i) = \text{argmin}_a \left[g(s_i, a) + J^*\left(f(s_i, a)\right) \right]$$

Value Iteration Example

- Mod

Hamilton-Jacobi-Bellman Equation

- Given J* and $\boldsymbol{\pi}^*$, we can use the HJB to verify our solution is optimal if it satisfies the HJB equation

$$0 = \min_{\mathbf{u}} \left[g(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) \right]$$

Linear Quadratic Regulator (LQR)

LQR

- Dynamic Programming solution without value iteration
- Achieves a closed form solution using quadratic costs

System

- Linear Time Invariant (LTI) system
- state x (n by 1)
- input u (m by 1)
- state matrix A (n by n)
- Input matrix B (n by m)

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

Stabilization Assumptions

- system is controllable

  $[B \ AB \ A^2B \ \dots \ A^{n-1}B]$

- infinite time
- has an equilibrium state $x = 0$
- Our goal is to drive the system to $x = 0$

Derivation - cost

- away from equilibrium
- large inputs to the system

Derivation - cost

- stage cost $g(x, u, t) = x^T Q x + u^T R u$
- Q, R are positive definite. In practice non-zero on diagonals

$$Q = \begin{matrix} q_1 & 0 & 0 \dots 0 \\ 0 & q_2 & 0 \dots 0 \\ 0 & 0 & q_3 .. 0 \\ .. & .. & .. \;.. 0 \\ 0 & 0 & 0 \;.. q_n \end{matrix} \qquad R = \begin{matrix} r_1 & 0 & 0 \dots 0 \\ 0 & r_2 & 0 \dots 0 \\ 0 & 0 & r_3 .. 0 \\ .. & .. & .. \;.. 0 \\ 0 & 0 & 0 \;.. r_m \end{matrix}$$

Derivation - from Hamilton Jacobi Bellman

$$0 = \min_{\mathbf{u}} \left[\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} + \frac{\partial J^*}{\partial \mathbf{x}} \left(\mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{u} \right) \right]$$

$$J^*(\mathbf{x}) = \mathbf{x}^T \mathbf{S} \mathbf{x}$$

$$\frac{\partial}{\partial \mathbf{u}} = 2\mathbf{u}^T \mathbf{R} + 2\mathbf{x}^T \mathbf{S} \mathbf{B} = 0$$

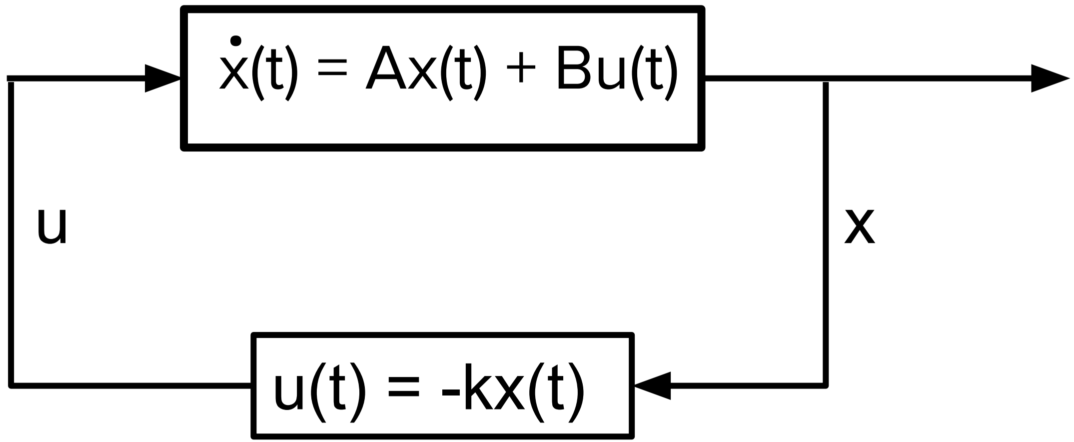
$$\mathbf{u}^* = -\mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} \mathbf{x} = -\mathbf{K} \mathbf{x}$$

Derivation - Algebraic Ricatti Equation (ARE)

$$0 = \min_{\mathbf{u}} \left[\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} + \frac{\partial J^*}{\partial \mathbf{x}} \left(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \right) \right]$$

$$0 = \mathbf{S}\mathbf{A} + \mathbf{A}^T \mathbf{S} - \mathbf{S}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T \mathbf{S} + \mathbf{Q}$$

feedback stabilization



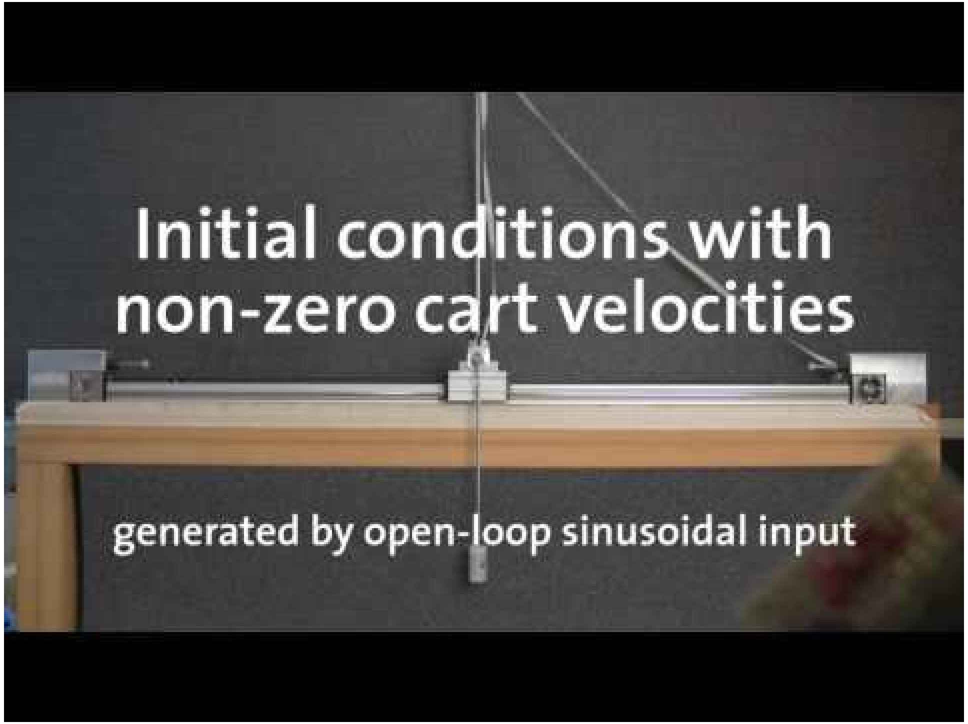
LQR insights

- For a quadratic cost the optimal solution is closed form
- Optimal controller has constant gain
- LQR picks poles for your system

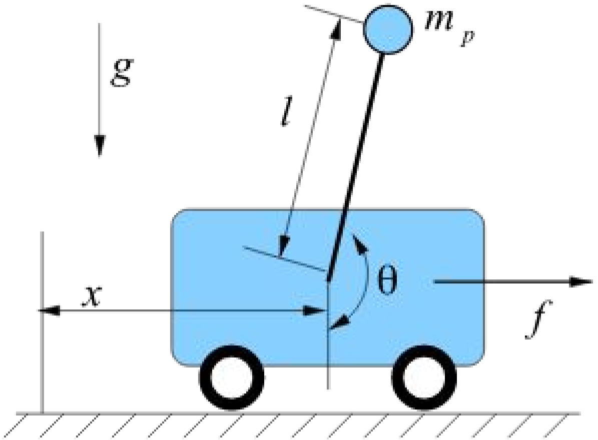
LQR - Stability and Robustness

- stable performance when linearity holds and system is controllable
- Robust in the face of model error
- Not Robust in the face of state error
    - Other methods such as H-infinity control exist

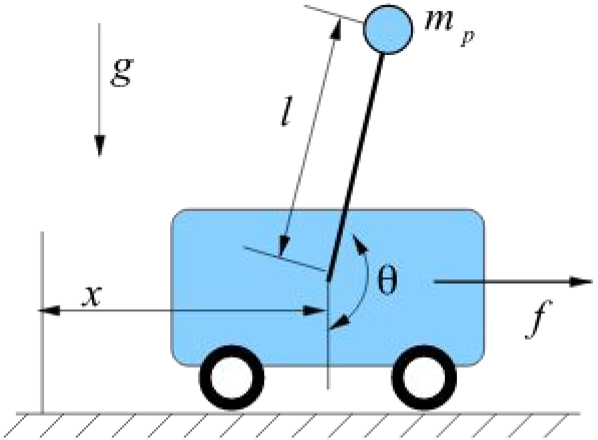
Example: The Cart-Pole System



Cart-pole system



- Position of cart: $$\mathbf{x}_1 = \begin{bmatrix} x \\ 0 \end{bmatrix}$$
- Position of mass on pole: $$\mathbf{x}_2 = \begin{bmatrix} x + l\sin\theta \\ -l\cos\theta \end{bmatrix}$$



$$T = \frac{1}{2}(m_c + m_p)\dot{x}^2 + m_p\dot{x}\dot{\theta}l\cos\theta + \frac{1}{2}m_p l^2\dot{\theta}^2$$
$$U = -m_p g l\cos\theta.$$

Lagrangian

$$L = T - U$$

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

$$(m_c + m_p)\ddot{x} + m_p l\ddot{\theta}\cos\theta - m_p l\dot{\theta}^2 \sin\theta = f$$
$$m_p l\ddot{x}\cos\theta + m_p l^2\ddot{\theta} + m_p g l\sin\theta = 0$$

Manipulator Equations

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

$$\mathbf{H}(\mathbf{q}) = \begin{bmatrix} m_c + m_p & m_p l\cos\theta \\ m_p l\cos\theta & m_p l^2 \end{bmatrix}, \quad \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix} 0 & -m_p l\dot{\theta}\sin\theta \\ 0 & 0 \end{bmatrix},$$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} 0 \\ m_p g l\sin\theta \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Linearizing

$$x^* = [0, \pi, 0, 0]^T$$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \approx \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*} (\mathbf{x} - \mathbf{x}^*) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right]_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*} (\mathbf{u} - \mathbf{u}^*)$$

$$\dot{\bar{\mathbf{x}}} = \mathbf{A}_{lin}\bar{\mathbf{x}} + \mathbf{B}_{lin}\bar{\mathbf{u}}.$$

Linearizing continued

$$\dot{\mathbf{x}} = \mathbf{A}_{lin}\bar{\mathbf{x}} + \mathbf{B}_{lin}\bar{\mathbf{u}}.$$

$$\mathbf{A}_{lin} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{H}^{-1}\frac{\partial \mathbf{G}}{\partial \mathbf{q}} + \sum_j \mathbf{H}^{-1}\frac{\partial \mathbf{B}_j}{\partial \mathbf{q}} u_j & -\mathbf{H}^{-1}\mathbf{C} \end{bmatrix}_{\mathbf{x}=\mathbf{x}^*,\mathbf{u}=\mathbf{u}^*}$$

$$\mathbf{B}_{lin} = \begin{bmatrix} \mathbf{0} \\ \mathbf{H}^{-1}\mathbf{B} \end{bmatrix}_{\mathbf{x}=\mathbf{x}^*,\mathbf{u}=\mathbf{u}^*}$$

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})_{\mathbf{x}=\mathbf{x}^*} = \mathbf{0}, \quad \left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}}\right]_{\mathbf{x}=\mathbf{x}^*} = \begin{bmatrix} 0 & 0 \\ 0 & -m_p g l \end{bmatrix}$$

Related Topics and Applications

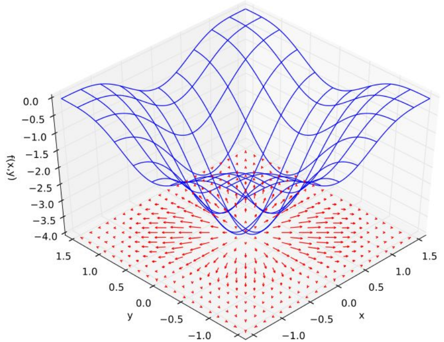
Policy Search

- Parametrize the system and search the parameter space for the best solution.
- All of the parameters go into a single vector α, and the controller is written as $\pi\_\alpha(x, t)$
  - Open loop controller: $u = \alpha\_n$, where $n = \text{floor}(t/dt)$
  - Feedback controller: $u = - K\_\alpha\ x$ (matrix K is written in terms of $\alpha1, \alpha2, \alpha3$, ... etc)
- Want to minimize the cost function $J^\wedge\alpha(x, t)$ for some initial condition
- There are two ways to optimize an open-loop trajectory (a trajectory that depends on time, not state): Shooting Methods and Direct Collocation.

Shooting Methods

- Keep shooting (simulating) values of α until goal is reached
- Define α as decision variables and evaluate J α(x0) through forward simulation
- Compute policy gradient ∂Jα(x0) / ∂α by
    - Backward Propagation Through Time (BPTT)
    - Real Time Recurrent Learning (RTRL)
- Locally optimal
- Cost function

$$J(\mathbf{x}_0) = \int_0^T g(\mathbf{x}(t), \mathbf{u}(t))dt, \quad \mathbf{x}(0) = \mathbf{x}_0.$$



Shooting Methods (continued)

- BPTT:
  - Integrate xdot = f(x, πα(x, t)) forward in time from 0 to T, starting from initial x(0)
  - Integrate $$-\dot{\mathbf{y}} = \mathbf{F}_{\mathbf{x}}^T \mathbf{y} - \mathbf{G}_{\mathbf{x}}^T$$
  - backward in time until $t = 0$, where
  - $$\mathbf{F}_{\mathbf{x}}(t) = \frac{\partial f}{\partial \mathbf{x}(t)} + \frac{\partial f}{\partial \mathbf{u}(t)} \frac{\partial \pi_\alpha}{\partial \mathbf{x}(t)}, \quad \mathbf{G}_{\mathbf{x}}(t) = \frac{\partial g}{\partial \mathbf{x}(t)} + \frac{\partial g}{\partial \mathbf{u}(t)} \frac{\partial \pi_\alpha}{\partial \mathbf{x}(t)},$$
  - evaluated at $\mathbf{x}(t), \mathbf{u}(t)$.
  -
  - − ydot = F^T_x y − G^T_x backward in time from T to 0
  - Y represents the sensitivity of J to change in x.
  - $$\frac{\partial J(\mathbf{x}_0)}{\partial \alpha} = \int_0^T dt \left[\mathbf{G}_\alpha^T - \mathbf{F}_\alpha^T \mathbf{y}\right]$$

RTRL

- $$J(\mathbf{x}_0) = \int_0^T g(\mathbf{x}, \mathbf{u}, t)dt.$$

$$\frac{d}{dt}\frac{\partial \mathbf{x}}{\partial \alpha} = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\frac{\partial \pi_\alpha}{\partial \mathbf{x}}\right]\frac{\partial \mathbf{x}}{\partial \alpha} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\frac{\partial \pi_\alpha}{\partial \alpha}$$

$$\dot{\mathbf{P}} = \mathbf{F_x P} + \mathbf{F}_\alpha \quad \mathbf{P}(0) = \mathbf{0}$$

$$P_{ij} = \frac{\partial x_i}{\partial \alpha_j}.$$

- By chain rule

$$\begin{aligned}\frac{\partial J}{\partial \alpha} &= \int_0^T dt \left[\frac{\partial g}{\partial \mathbf{x}(t)}\frac{\partial \mathbf{x}(t)}{\partial \alpha} + \frac{\partial g}{\partial \mathbf{u}(t)}\frac{\partial \pi_\alpha}{\partial \alpha} + \frac{\partial g}{\partial \mathbf{u}(t)}\frac{\partial \pi}{\partial \mathbf{x}(t)}\frac{\partial \mathbf{x}(t)}{\partial \alpha}\right] \\ &= \int_0^T dt \left[\mathbf{G_x P} + \mathbf{G}_\alpha\right]\end{aligned}$$

LQR Trajectory Stabilization (1 of 4)

- Problem: There isn't a guarantee that simulations of the optimized trajectories won't diverge from the planned trajectories (e.g. different time step size, modeling errors… etc)
- Given a system, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}),$ we want to express as $\dot{x}(t) = A\,x(t) + B\,u(t)$
- Linearize around a random point (x_0, u_0) and Taylor expand

$$\dot{\mathbf{x}} \approx \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x} - \mathbf{x}_0) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_0) = \mathbf{c} + \mathbf{A}(\mathbf{x} - \mathbf{x}_0) + \mathbf{B}(\mathbf{u} - \mathbf{u}_0)$$

- Change coordinate system

$$\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_0(t), \quad \bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_0(t),$$

LQR Trajectory Stabilization (2 of 4)

- We use x_0(t) and u_0(t) because they are a solution to the dynamics.

$$\dot{\bar{\mathbf{x}}}(t) = \dot{\mathbf{x}}(t) - \dot{\mathbf{x}}_0(t) = \dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t)),$$

- Remember our system is $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}),$
- And our Taylor expansion is
- $\dot{\mathbf{x}} \approx \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x} - \mathbf{x}_0) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_0) = \mathbf{c} + \mathbf{A}(\mathbf{x} - \mathbf{x}_0) + \mathbf{B}(\mathbf{u} - \mathbf{u}_0)$
- Now we have

$$\begin{aligned}\dot{\bar{\mathbf{x}}}(t) =& \frac{\partial \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t))}{\partial \mathbf{x}}(\mathbf{x}(t) - \mathbf{x}_0(t)) + \frac{\partial \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t))}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_0(t)) \\ =& \mathbf{A}(t)\bar{\mathbf{x}}(t) + \mathbf{B}(t)\bar{\mathbf{u}}(t).\end{aligned}$$

LQR Trajectory Stabilization (3 of 4)

- This means our system depends linearly on time
- Linear Time Varying system LTV
- Principles on linear systems still apply like superposition
- And we can do LQR

LQR Trajectory Stabilization: (4 of 4)

- Minimize the cost function

$$J(\mathbf{x}_0, 0) = \bar{\mathbf{x}}(t_f)^T \mathbf{Q}_f \bar{\mathbf{x}}(t_f) + \int_0^T dt \left[\bar{\mathbf{x}}(t)^T \mathbf{Q} \bar{\mathbf{x}}(t) + \bar{\mathbf{u}}(t)^T \mathbf{R} \bar{\mathbf{u}}(t) \right].$$

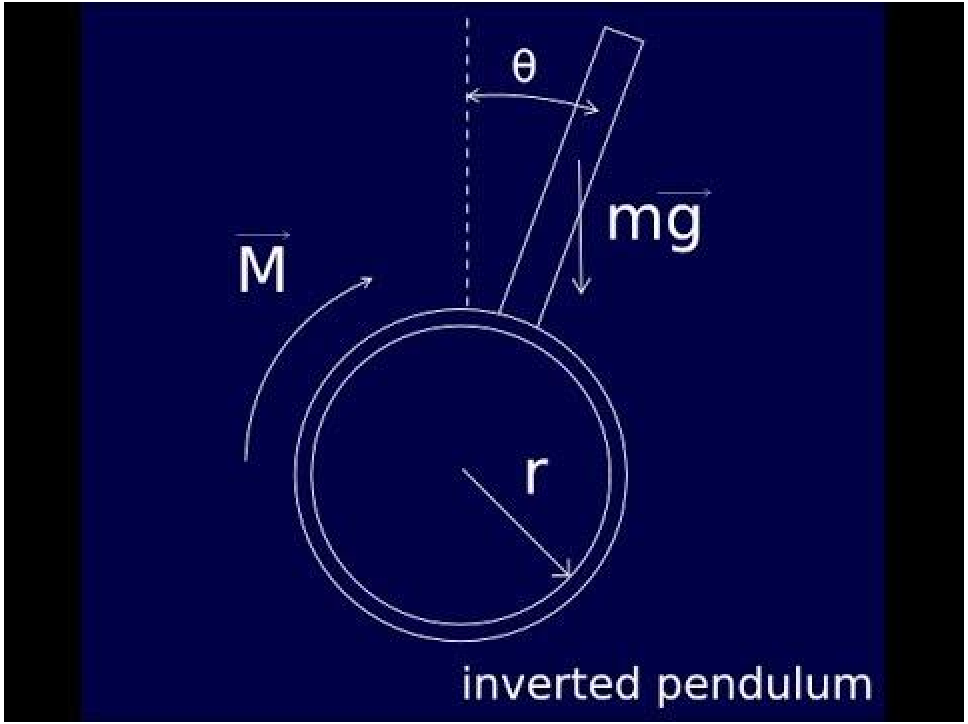
- Penalizes the system at t for being away from x_0(t).
- Solve the Hamilton-Jacobi-Bellman equation and the boundary conditions to get LTV LQR

Iterative LQR

- Can replace shooting method
- Given the cost function used in the shooting method RTRL, we can Taylor expand

$$g(\mathbf{x},\mathbf{u}) \approx g(\mathbf{x}_0,\mathbf{u}_0) + \frac{\partial g}{\partial \mathbf{x}}\bar{\mathbf{x}} + \frac{\partial g}{\partial \mathbf{u}}\bar{\mathbf{u}} + \frac{1}{2}\bar{\mathbf{x}}^T\frac{\partial^2 g}{\partial \mathbf{x}^2}\bar{\mathbf{x}} + \bar{\mathbf{x}}\frac{\partial^2 g}{\partial \mathbf{x}\partial \mathbf{u}}\bar{\mathbf{u}} + \frac{1}{2}\bar{\mathbf{u}}^T\frac{\partial^2 g}{\partial \mathbf{u}^2}\bar{\mathbf{u}}.$$

- Do the time-varying linearization around the normal trajectory x_0(t) and u_0(t)
- Rewards the system for going to minimum cost function, not stabilizing around x_0(t) and u_0(t). Stabilize around x_d(t) that minimizes cost.
- Each iteration, compute a new u_0(t) from initial conditions


θ
$\overrightarrow{mg}$
$\overrightarrow{M}$
r
inverted pendulum