ME 599/699 Robot Modeling & Control
Fall 2021

Optimal Control

Hasan A. Poonawala

Department of Mechanical Engineering
University of Kentucky

Email: hasan.poonawala@uky.edu
Web: https://www.engr.uky.edu/~hap

Optimal Control

In continuous time, we have

$$\begin{aligned} \text{min} \quad & J(q(t), u(t)) \\ \text{subject to} \quad & q(t) \text{ satisfies dynamics and state constraints} \\ & u(t) \text{ satisfies input constraints} \end{aligned}$$

We may also formulate discrete time versions of this problem.

Linear Quadratic Regulator

For optimal control problems where

- time is discrete,
- the dynamics are linear, and
- the cost function is quadratic in state and control,

the optimal control problem may be solved in a straightforward way.

These slides are inspired by Sergey Levine's slides.

Linear Quadratic Regulator

At each time $t \in \{0, 1, 2, \ldots, T\}$, we have

$$\mathbf{x}_{t+1} = A_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + a_t; \quad c_t(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t$$

Consider a finite time horizon $t \in \{0, 1, 2, \ldots, T\}$.
Let

$$J = \sum_{t=0}^{T} c_t(x_t, u_t)$$

Focus on T

At time T, we have only one decision to make: pick u_T.

Focus on T

At time T, we have only one decision to make: pick u_T.

The cost of doing so is exactly $c_T(\mathbf{x}_T, \mathbf{u}_T)$

Focus on T

At time T, we have only one decision to make: pick u_T.

The cost of doing so is exactly $c_T(\mathbf{x}_T, \mathbf{u}_T)$

The cost for the first $T-1$ time steps are some value that is effectively constant at time T, so that the total cost will be $\mathbf{Q}_T(\mathbf{x}_T, \mathbf{u}_T)$

$$\mathbf{Q}_T(\mathbf{x}_T, \mathbf{u}_T) = \mathrm{const} + \frac{1}{2}\begin{bmatrix}\mathbf{x}_T \\ \mathbf{u}_T\end{bmatrix}^T \mathbf{C}_T \begin{bmatrix}\mathbf{x}_T \\ \mathbf{u}_T\end{bmatrix} + \begin{bmatrix}\mathbf{x}_T \\ \mathbf{u}_T\end{bmatrix}^T \mathbf{c}_T$$

Optimize at T

To find the best $\mathbf{u}_T$, we minimize that expression.

It's gradient w.r.t. $\mathbf{u}_T$ is

$$\nabla_{\mathbf{u}_T} \mathbf{Q}_T(x_T, u_T) = x_T^T \mathbf{C}_{\mathbf{x}_T, \mathbf{u}_T} + u_T^T \mathbf{C}_{\mathbf{u}_T, \mathbf{u}_T} + \mathbf{c}_{\mathbf{u}_T}^T, \text{ where}$$

$$\mathbf{C}_T = \begin{bmatrix} \mathbf{C}_{\mathbf{x}_T, \mathbf{x}_T} & \mathbf{C}_{\mathbf{x}_T, \mathbf{u}_T} \\ \mathbf{C}_{\mathbf{x}_T, \mathbf{u}_T} & \mathbf{C}_{\mathbf{u}_T, \mathbf{u}_T} \end{bmatrix}, \quad \mathbf{c}_T = \begin{bmatrix} \mathbf{c}_{\mathbf{x}_T} \\ \mathbf{c}_{\mathbf{u}_T} \end{bmatrix}.$$

Optimize at T

To find the best $\mathbf{u}_T$, we minimize that expression.

It's gradient w.r.t. $\mathbf{u}_T$ is

$$\nabla_{\mathbf{u}_T}\mathbf{Q}_T(x_T, u_T) = x_T^T\mathbf{C}_{\mathbf{x}_T,\mathbf{u}_T} + u_T^T\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T} + \mathbf{c}_{\mathbf{u}_T}^T, \text{ where}$$

$$\mathbf{C}_T = \begin{bmatrix} \mathbf{C}_{\mathbf{x}_T,\mathbf{x}_T} & \mathbf{C}_{\mathbf{x}_T,\mathbf{u}_T} \\ \mathbf{C}_{\mathbf{x}_T,\mathbf{u}_T} & \mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T} \end{bmatrix}, \quad \mathbf{c}_T = \begin{bmatrix} \mathbf{c}_{\mathbf{x}_T} \\ \mathbf{c}_{\mathbf{u}_T} \end{bmatrix}.$$

Setting $\nabla_{u_T} Q_T(x_T, u_T) = 0$ we obtain

$$\mathbf{u}_T = -\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}^{-1}\left(\mathbf{C}_{\mathbf{x}_T,\mathbf{u}_T}\mathbf{x}_T + \mathbf{c}_{\mathbf{u}_T}\right) = \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T,$$

which is a linear (well, affine) feedback control.

Cutting to the Chase

- To cut a long story short,

$$\mathbf{Q}_T(\mathbf{x}_T, \mathbf{u}_T) = \mathbf{Q}_T(\mathbf{x}_T, \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T) = V(\mathbf{x}_T) = \mathbf{x}_T^T\mathbf{V}_T\mathbf{x}_T + \mathbf{x}_T^T\mathbf{v}_T,$$

  for some appropriate matrix $\mathbf{V}_T$ and $\mathbf{v}_T$ that depends on the problem's parameters.

Cutting to the Chase

- To cut a long story short,

$$\mathbf{Q}_T(\mathbf{x}_T, \mathbf{u}_T) = \mathbf{Q}_T(\mathbf{x}_T, \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T) = V(\mathbf{x}_T) = \mathbf{x}_T^T \mathbf{V}_T \mathbf{x}_T + \mathbf{x}_T^T \mathbf{v}_T,$$

  for some appropriate matrix $\mathbf{V}_T$ and $\mathbf{v}_T$ that depends on the problem's parameters.

- Because the dynamics are linear, and costs are quadratic, the same thing repeats at $t = T - 1$

$$\begin{aligned}
\mathbf{Q}_{T-1}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) &= \text{const} + c_{T-1}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) + V(\mathbf{x}_T) \\
&= \text{const} + c_{T-1}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) \\
&\qquad + V\left(A_{T-1}\begin{bmatrix}\mathbf{x}_{T-1} \\ \mathbf{u}_{T-1}\end{bmatrix} + a_{T-1}\right) \\
&= \text{Quadratic}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1})
\end{aligned}$$

Cutting to the Chase

- To cut a long story short,

$$\mathbf{Q}_T(\mathbf{x}_T, \mathbf{u}_T) = \mathbf{Q}_T(\mathbf{x}_T, \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T) = V(\mathbf{x}_T) = \mathbf{x}_T^T\mathbf{V}_T\mathbf{x}_T + \mathbf{x}_T^T\mathbf{v}_T,$$

  for some appropriate matrix $\mathbf{V}_T$ and $\mathbf{v}_T$ that depends on the problem's parameters.
- Because the dynamics are linear, and costs are quadratic, the same thing repeats at $t = T - 1$

$$\begin{aligned}\mathbf{Q}_{T-1}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) &= \mathrm{const} + c_{T-1}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) + V(\mathbf{x}_T) \\ &= \mathrm{const} + c_{T-1}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) \\ &\qquad + V\left(A_{T-1}\begin{bmatrix}\mathbf{x}_{T-1} \\ \mathbf{u}_{T-1}\end{bmatrix} + a_{T-1}\right) \\ &= \mathrm{Quadratic}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1})\end{aligned}$$

- The optimal control at $t = T - 1$ will be linear, and so on

Cutting to the Chase

- To cut a long story short,

$$\mathbf{Q}_T(\mathbf{x}_T, \mathbf{u}_T) = \mathbf{Q}_T(\mathbf{x}_T, \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T) = V(\mathbf{x}_T) = \mathbf{x}_T^T \mathbf{V}_T \mathbf{x}_T + \mathbf{x}_T^T \mathbf{v}_T,$$

  for some appropriate matrix $\mathbf{V}_T$ and $\mathbf{v}_T$ that depends on the problem's parameters.
- Because the dynamics are linear, and costs are quadratic, the same thing repeats at $t = T - 1$

$$\begin{aligned}
\mathbf{Q}_{T-1}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) &= \text{const} + c_{T-1}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) + V(\mathbf{x}_T) \\
&= \text{const} + c_{T-1}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) \\
&\qquad + V\left(A_{T-1}\begin{bmatrix}\mathbf{x}_{T-1} \\ \mathbf{u}_{T-1}\end{bmatrix} + a_{T-1}\right) \\
&= \text{Quadratic}(\mathbf{x}_{T-1}, \mathbf{u}_{T-1})
\end{aligned}$$

- The optimal control at $t = T - 1$ will be linear, and so on
- This nice structure persists till $t = 0$

Optimal Control

This procedure nicely illustrates some of the core ideas

1. Solve for the best control by moving backwards in time

Optimal Control

This procedure nicely illustrates some of the core ideas

1. Solve for the best control by moving backwards in time
2. By building up an estimate of the cost-to-go (V)

Optimal Control

This procedure nicely illustrates some of the core ideas

1. Solve for the best control by moving backwards in time
2. By building up an estimate of the cost-to-go (V)
3. The function $Q_t(\mathbf{x}_t, \mathbf{u}_t)$ is known as the Q-function in reinforcement learning

Optimal Control

This procedure nicely illustrates some of the core ideas

1. Solve for the best control by moving backwards in time
2. By building up an estimate of the cost-to-go (V)
3. The function $Q_t(\mathbf{x}_t, \mathbf{u}_t)$ is known as the Q-function in reinforcement learning
4. V is the value function (we minimize, RL maximizes)

Optimal Control

- The core method was a constructive approach:
  Solve quadratic optimizations at each step to build V_T

Optimal Control

- The core method was a constructive approach:
  Solve quadratic optimizations at each step to build V_T
- Instead, some approaches compute $V_T/V(t)$ directly
  (Hamilton-Jacobi-Belmlman equations)

Optimal Control

- The core method was a constructive approach:
  Solve quadratic optimizations at each step to build V_T
- Instead, some approaches compute $V_T/V(t)$ directly
  (Hamilton-Jacobi-Belmlman equations)
- These methods require knowing dynamics and reward functions

Reinforcement Learning

- What if we don't know cost $\mathbf{c}_T$, dynamics $x_{T+1} = f(x_T, u_T)$?

Reinforcement Learning

- What if we don't know cost $\mathbf{c}_T$, dynamics $x_{T+1} = f(x_T, u_T)$?
- We can't 'solve' for control from known models

Reinforcement Learning

- What if we don't know cost $\mathbf{c}_T$, dynamics $x_{T+1} = f(x_T, u_T)$?
- We can't 'solve' for control from known models
- We must instead learn from a stream of experience data

Reinforcement Learning

- What if we don't know cost $\mathbf{c}_T$, dynamics $x_{T+1} = f(x_T, u_T)$?
- We can't 'solve' for control from known models
- We must instead learn from a stream of experience data
- Main challenge is in trading-off learning and optimizing (exploration-exploitation trade-off)

RL Basics

- Two major steps in learning from experience:

RL Basics

- Two major steps in learning from experience:
  - Policy Evaluation: How do we evaluate a choice for actions (called policy π)?

RL Basics

- Two major steps in learning from experience:
  - Policy Evaluation: How do we evaluate a choice for actions (called policy π)?
  - Policy Improvement: How do we improve the policy?

RL Basics

- Two major steps in learning from experience:
  - Policy Evaluation: How do we evaluate a choice for actions (called policy π)?
  - Policy Improvement: How do we improve the policy?
- Two basic philosophies:

RL Basics

- Two major steps in learning from experience:
  - Policy Evaluation: How do we evaluate a choice for actions (called policy π)?
  - Policy Improvement: How do we improve the policy?
- Two basic philosophies:
  - Model-based: Build models of dynamics and rewards form data, ‘solve’ to improve/evaluate at the same time.

RL Basics

- Two major steps in learning from experience:
  - Policy Evaluation: How do we evaluate a choice for actions (called policy π)?
  - Policy Improvement: How do we improve the policy?
- Two basic philosophies:
  - Model-based: Build models of dynamics and rewards form data, 'solve' to improve/evaluate at the same time.
  - Model-free: Maintain policy π and V using data

RL Algorithms

Terms you will come across

- Policy Iteration: Evaluate policy π to get $V(\pi)$, improve policy using $V(\pi)$
  (many variations for both steps)

RL Algorithms

Terms you will come across

- Policy Iteration: Evaluate policy π to get $V(\pi)$, improve policy using $V(\pi)$
  (many variations for both steps)
- Value iteration: Learn value function V directly

RL Algorithms

Terms you will come across

- Policy Iteration: Evaluate policy π to get $V(\pi)$, improve policy using $V(\pi)$
  (many variations for both steps)
- Value iteration: Learn value function V directly
- Monte Carlo (REINFORCE, AlphaGo)

RL Algorithms

Terms you will come across

- Policy Iteration: Evaluate policy π to get $V(\pi)$, improve policy using $V(\pi)$
  (many variations for both steps)
- Value iteration: Learn value function V directly
- Monte Carlo (REINFORCE, AlphaGo)
- Temporal differences (Q-Learning / SARSA)

RL Algorithms

Terms you will come across

- Policy Iteration: Evaluate policy π to get $V(\pi)$, improve policy using $V(\pi)$
  (many variations for both steps)
- Value iteration: Learn value function V directly
- Monte Carlo (REINFORCE, AlphaGo)
- Temporal differences (Q-Learning / SARSA)
- Policy gradients (PPO, SAC)

RL Algorithms

Terms you will come across

- Policy Iteration: Evaluate policy π to get $V(\pi)$, improve policy using $V(\pi)$
  (many variations for both steps)
- Value iteration: Learn value function V directly
- Monte Carlo (REINFORCE, AlphaGo)
- Temporal differences (Q-Learning / SARSA)
- Policy gradients (PPO, SAC)
- Optimization (TRPO, iLQR)

Deep RL and Robotics

- Model-free Deep RL helps escape modeling challenges in robot control

Deep RL and Robotics

- Model-free Deep RL helps escape modeling challenges in robot control
- V, Q, π are deep neural networks (challenging in continuous spaces)

Deep RL and Robotics

- Model-free Deep RL helps escape modeling challenges in robot control
- V, Q, π are deep neural networks (challenging in continuous spaces)
- Zoo of approaches (PPO, SAC, MBPO, DDPG)

Deep RL and Robotics

- Model-free Deep RL helps escape modeling challenges in robot control
- V, Q, π are deep neural networks (challenging in continuous spaces)
- Zoo of approaches (PPO, SAC, MBPO, DDPG)
- Learn in sim, fine-tune in reality, or robustify

Deep RL and Robotics

- Model-free Deep RL helps escape modeling challenges in robot control
- V, Q, π are deep neural networks (challenging in continuous spaces)
- Zoo of approaches (PPO, SAC, MBPO, DDPG)
- Learn in sim, fine-tune in reality, or robustify
- Most successful approaches use low-level position-based control (impedance or otherwise) on position-based tasks

Caution

- State-of-the-Art is extremely hard to evaluate:

Caution

- State-of-the-Art is extremely hard to evaluate:
  - new papers out every week

Caution

- State-of-the-Art is extremely hard to evaluate:
  - new papers out every week
  - videos/sims are always impressive

Caution

- State-of-the-Art is extremely hard to evaluate:
  - new papers out every week
  - videos/sims are always impressive
  - same trick as robotics: show only the few working cases

Caution

- State-of-the-Art is extremely hard to evaluate:
  - new papers out every week
  - videos/sims are always impressive
  - same trick as robotics: show only the few working cases
  - often rely on well-tuned low-level controllers or unrealistic amounts of training data

Caution

- State-of-the-Art is extremely hard to evaluate:
  - new papers out every week
  - videos/sims are always impressive
  - same trick as robotics: show only the few working cases
  - often rely on well-tuned low-level controllers or unrealistic amounts of training data
- Most papers are opaque about how much human engineering is involved

Caution

- State-of-the-Art is extremely hard to evaluate:
  - new papers out every week
  - videos/sims are always impressive
  - same trick as robotics: show only the few working cases
  - often rely on well-tuned low-level controllers or unrealistic amounts of training data
- Most papers are opaque about how much human engineering is involved
- My opinion: use to choose controller, not to design control

Caution

- State-of-the-Art is extremely hard to evaluate:
  - new papers out every week
  - videos/sims are always impressive
  - same trick as robotics: show only the few working cases
  - often rely on well-tuned low-level controllers or unrealistic amounts of training data
- Most papers are opaque about how much human engineering is involved
- My opinion: use to choose controller, not to design control
- My lab: learn NN models from data, design correct controllers for such models