Linear Quadratic Regulator

Sanjiban Choudhury

TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle
Different control laws

1. Bang-bang control
2. PID control
3. Pure-pursuit control
4. Lyapunov control
5. LQR
6. MPC
Recap of controllers

PID / Pure pursuit: Worked well, no provable guarantees

Lyapunov: Provable stability, *convergence rate depends on gains*
# Table of controllers

<table>
<thead>
<tr>
<th>Control Law</th>
<th>Uses model</th>
<th>Stability Guarantee</th>
<th>Minimize Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID</td>
<td>$u = K_p e + ...$</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Pure Pursuit</td>
<td>$u = \tan^{-1}\left(\frac{2B \sin \alpha}{L}\right)$</td>
<td>Circular arcs</td>
<td>Yes - with assumptions</td>
</tr>
<tr>
<td>Lyapunov</td>
<td>$u = \tan^{-1}\left(-\frac{k_1 e_{\alpha} B}{\theta e \sin \theta e} - \frac{B}{V} k_2 \theta e\right)$</td>
<td>Non-linear</td>
<td>Yes</td>
</tr>
</tbody>
</table>

---
Is stability enough?

\[ \lim_{t \to \infty} e(t) = 0 \]
Is *stability* enough of a guarantee?
Is **stability** enough of a guarantee?
Is **stability** enough of a guarantee?

Control action changes abruptly - why is this bad?
Is **stability** enough of a guarantee?

What if we just choose really small gains?
Is stability enough of a guarantee?

What if we just choose really small gains?
Is **stability** enough of a guarantee?

What if we just choose really small gains?

Stability guarantees that the error will go to zero ... but can take arbitrary long time
Question:
How do we trade-off both driving error to zero AND keeping control action small?
Key Idea:

Turn the problem into an optimization

$$\min_{u(t)} \int_0^\infty \left( w_1 e(t)^2 + w_2 u(t)^2 \right) dt$$
Optimal Control
A fundamental framework:

Linear Quadratic Regulator

Trivia! :) (from http://www.uta.edu/utari/acs/history.htm)

In 1960 three major papers were published by R. Kalman and coworkers...
1. One of these [Kalman and Bertram 1960], publicized the vital work of Lyapunov in the time-domain control of nonlinear systems.
2. The next [Kalman 1960a] discussed the optimal control of systems, providing the design equations for the linear quadratic regulator (LQR).
3. The third paper [Kalman 1960b] discussed optimal filtering and estimation theory, providing the design equations for the discrete Kalman filter.
LQR flying RC helicopters

(Excellent work by Pieter Abeel et al. https://people.eecs.berkeley.edu/~pabbeel/autonomous_helicopter.html)
LQR flying RC helicopters

(EExcellent work by Pieter Abeel et al. [https://people.eecs.berkeley.edu/~pabbeel/autonomous_helicopter.html](https://people.eecs.berkeley.edu/~pabbeel/autonomous_helicopter.html))
A fundamental framework:

Linear Quadratic Regulator
The Linear Quadratic Regulator (LQR) Problem

Given:
The Linear Quadratic Regulator (LQR) Problem

Given:

1. Linear dynamical system

\[ x_{t+1} = Ax_t + Bu_t \]
The Linear Quadratic Regulator (LQR) Problem

Given:

1. **Linear** dynamical system

   \[ x_{t+1} = Ax_t + Bu_t \]

2. A reference state which we are **regulating** around

   \[ x_{ref} = 0 \]
The Linear Quadratic Regulator (LQR) Problem

Given:

1. **Linear** dynamical system

\[ x_{t+1} = Ax_t + Bu_t \]

2. A reference state which we are **regulating** around

\[ x_{\text{ref}} = 0 \]

3. A **quadratic** cost function to minimize

\[
\begin{align*}
c(x_t, u_t) &= (x_t - x_{\text{ref}})^T Q (x_t - x_{\text{ref}}) + u_t^T R u_t \\
&= x_t^T Q x_t + u_t^T R u_t
\end{align*}
\]
The Linear Quadratic Regulator (LQR) Problem

Given:

1. **Linear** dynamical system
   \[ x_{t+1} = Ax_t + Bu_t \]

2. A reference state which we are **regulating** around
   \[ x_{ref} = 0 \]

3. A **quadratic** cost function to minimize
   \[ c(x_t, u_t) = (x_t - x_{ref})^T Q (x_t - x_{ref}) + u_t^T Ru_t \]
   \[ = x_t^T Q x_t + u_t^T R u_t \]

**Goal:** Compute control actions to minimize cumulative cost (value)

\[ J = \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t \]
Example: Inverted Pendulum
Example: Inverted Pendulum

Equations of motion

\[ ml^2 \ddot{\theta} - mgl \sin \theta = \tau \]
Example: Inverted Pendulum

Equations of motion

\[ ml^2 \ddot{\theta} - mgl \sin \theta = \tau \]

\[ \ddot{\theta} = \frac{g}{l} \sin \theta + \frac{1}{ml^2} \tau \]
Example: Inverted Pendulum

Equations of motion

\[ ml^2 \ddot{\theta} - mgl \sin \theta = \tau \]

\[ \ddot{\theta} = \frac{g}{l} \sin \theta + \frac{1}{ml^2} \tau \]

\[ \approx \frac{g}{l} \theta + \frac{1}{ml^2} \tau \]
Example: Inverted Pendulum

Equations of motion

\[ ml^2 \ddot{\theta} - mgl \sin \theta = \tau \]

\[ \ddot{\theta} = \frac{g}{l} \sin \theta + \frac{1}{ml^2} \tau \]

\[ \approx \frac{g}{l} \theta + \frac{1}{ml^2} \tau \]

\[
\begin{bmatrix}
\theta \\
\dot{\theta}
\end{bmatrix}
\begin{bmatrix}
1 + \frac{1}{2} \frac{g}{l} \Delta t^2 \\
\frac{g}{l} \Delta t
\end{bmatrix}
\begin{bmatrix}
\Delta t
\end{bmatrix}
\begin{bmatrix}
\theta \\
\dot{\theta}
\end{bmatrix}
+ \begin{bmatrix}
\frac{1}{2} \Delta t^2 \\
\Delta t
\end{bmatrix}
\frac{\tau}{ml^2}
\]

A

B
Get to $(0,0)$ while minimizing cost

\[ J = \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T Ru_t \]
Get to \((0,0)\) while minimizing cost

\[
J = \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t
\]
Get to \((0,0)\) while minimizing cost

\[ J = \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t \]
Get to \((0,0)\) while minimizing cost

\[
J = \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t
\]
Get to (0,0) while minimizing cost

\[ J = \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t \]
Get to (0,0) while minimizing cost

\[ J = \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t \]
Get to $(0,0)$ while minimizing cost

$$J = \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t$$
Get to (0,0) while minimizing cost

\[ J = \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t \]
Observation: Cost-to-go is not uniform

- Easier to be on this axis
- Harder to be on this axis

(0, 0)
How do we solve for controls?
How do we solve for controls?

Dynamic programming to the rescue!
How do we solve for controls?

Dynamic programming to the rescue!

Recall the Bellman function that relates value at consecutive time steps

\[ J(x_t, t) = \min_{u_t} c(x_t, u_t) + J(x_{t+1}, t + 1) \]

\[ = \min_{u_t} x_t^T Q x_t + u_t^T R u_t + J(x_{t+1}, t + 1) \]
How do we solve for controls?

**Dynamic programming** to the rescue!

Start from timestep T-1 and solve backwards
How do we solve for controls?

**Dynamic programming to the rescue!**

Start from timestep T-1 and solve backwards
How do we solve for controls?

**Dynamic programming to the rescue!**

Start from timestep T-1 and solve backwards

\[ x_{T-1} \]

\[ x_{T-2} \]

\[ x_{T-3} \]

\[ \text{Start from timestep T-1 and solve backwards} \]
How do we solve for controls?

Dynamic programming to the rescue!

Start from timestep T-1 and solve backwards

\[ \begin{align*}
  x_{T-2} &\rightarrow x_{T-1} \\
  u_{T-2} &\rightarrow x_{ref}
\end{align*} \]
How do we solve for controls?

Dynamic programming to the rescue!

Start from timestep T-1 and solve backwards
Last time step $T-1$
We have only 1 term in the cost function

\[ J(x_{T-1}, u_{T-1}) = \min_{u_T} x_{T-1}^T Q x_{T-1} + u_{T-1}^T R u_{T-1} \]
Last time step $T-1$

We have only 1 term in the cost function

$$J(x_{T-1}, u_{T-1}) = \min_{u_T} x_{T-1}^T Q x_{T-1} + u_{T-1}^T R u_{T-1}$$

To minimize cost, set control to 0

$$u_{T-1} = 0$$
Last time step $T-1$

We have only 1 term in the cost function

$$J(x_{T-1}, u_{T-1}) = \min_{u_T} x_{T-1}^T Q x_{T-1} + u_{T-1}^T R u_{T-1}$$

To minimize cost, set control to 0

$$u_{T-1} = 0$$

The cost function is a quadratic

$$J(x_{T-1}, u_{T-1}) = x_{T-1}^T Q x_{T-1}$$

$$= x_{T-1}^T V_{T-1} x_{T-1}$$

(this is a value matrix)
Last time step T-1

We have only 1 term in the cost function

$$J(x_{T-1}, u_{T-1}) = \min_{u_T} x_{T-1}^T Q x_{T-1} + u_{T-1}^T R u_{T-1}$$

To minimize cost, set control to 0

$$u_{T-1} = 0$$

The cost function is a quadratic

$$J(x_{T-1}, u_{T-1}) = x_{T-1}^T Q x_{T-1}$$

$$= x_{T-1}^T V_{T-1} x_{T-1}$$

(this is a value matrix)
Last time step $T-2$
Last time step $T-2$

$$J(x_{T-2}, T - 2) = \min_{u_{T-2}} c(x_{T-2}, u_{T-2}) + J(x_{T-1}, T - 1)$$
Last time step T-2

\[ J(x_{T-2}, T - 2) = \min_{u_{T-2}} c(x_{T-2}, u_{T-2}) + J(x_{T-1}, T - 1) \]

\[ = \min_{u_{T-2}} x_{T-2}^T Q x_{T-2} + u_{T-2}^T R u_{T-2} + x_{T-1}^T V_{T-1} x_{T-1} \]
Last time step $T-2$

\[ J(x_{T-2}, T-2) = \min_{u_{T-2}} c(x_{T-2}, u_{T-2}) + J(x_{T-1}, T-1) \]

\[ = \min_{u_{T-2}} x_{T-2}^T Q x_{T-2} + u_{T-2}^T R u_{T-2} + x_{T-1}^T V_{T-1} x_{T-1} \]

Solve for control at timestep $T-1$
Last time step \( T-2 \)

\[
J(x_{T-2}, T - 2) = \min_{u_{T-2}} c(x_{T-2}, u_{T-2}) + J(x_{T-1}, T - 1)
\]

\[
= \min_{u_{T-2}} x_{T-2}^T Q x_{T-2} + u_{T-2}^T R u_{T-2} + x_{T-1}^T V_{T-1} x_{T-1}
\]

Solve for control at timestep \( T-1 \)

\[
u_{T-2} = -(R + B^T V_{T-1} B)^{-1} B^T V_{T-1} A x_{T-2}
\]

Observation: Control law is linear!
Key insight: Value function is always quadratic

Plug back control in the value function (cumulative cost)
Key insight: Value function is always quadratic

Plug back control in the value function (cumulative cost)

\[ J(x_{T-2}, T - 2) = x_{T-2}^T(Q + K_{T-2}^T R K_{T-2} + (A + B K_{T-2})^T V_{T-1}(A + B K_{T-2})) x_{T-2} \]

\[ V_{T-2} \]
Key insight: Value function is always quadratic

Plug back control in the value function (cumulative cost)

\[
J(x_{T-2}, T - 2) = x_{T-2}^T(Q + K_{T-2}^TRK_{T-2} + (A + BK_{T-2})^TV_{T-1}(A + BK_{T-2}))x_{T-2}
\]

Quadratic remains quadratic
We can derive this relation at **ALL time steps**

\[ K_t = -(R + B^T V_{t+1} B)^{-1} B^T V_{t+1} A \]

\[ V_t = Q + K_t^T R K_t + (A + B K_t)^T V_{t+1} (A + B K_t) \]

| Current cost | Action cost | Closed loop dynamics | Future value func | Closed loop dynamics |
The LQR algorithm

Algorithm OptimalValue($A, B, Q, R, t, T$)

if $t = T - 1$ then
  return $Q$
end

else

  $V_{t+1} = \text{OptimalValue}(A, B, Q, R, t + 1, T)$

  $K_t = -(B^T V_{t+1} B + R)^{-1} B^T V_{t+1} A$

  return $V_t = Q + K_t^T R K_t + (A + BK_t)^T V_{t+1} (A + BK_t)$

end

(Courtesy Drew Bagnell)
Contours of value function (T-1)

\[ V_{T-1} = Q \]
Contours of value function \((T-2)\)

\[
V_{T-2} = Q + K^T R K \\
+ (A + BK)^T V_{T-1}(A + BK)
\]
Contours of value function (many steps)
How does the value function evolve?

Easier to be on this axis

Harder to be on this axis
What if my time horizon is very very very large?
Convergence of value iteration
Convergence of value iteration

Theorem: If the system is stabilizable, then the value $V$ will converge
Convergence of value iteration

**Theorem:** If the system is stabilizable, then the value $V$ will converge

$$V = Q + K^T R K + (A + B K)^T V (A + B K)$$

$$K = -(R + B^T V B)^{-1} B^T V A$$

Discrete Algebraic Ricatti Equation (DARE)
Convergence of value iteration

Theorem: If the system is stabilizable, then the value $V$ will converge

$$V = Q + K^T R K + (A + B K)^T V (A + B K)$$

$$K = - (R + B^T V B)^{-1} B^T V A$$

Discrete Algebraic Ricatti Equation (DARE)

How do I solve? Can iterate over $V$ / use eigen value decomposition [1]

Type into MATLAB:  `dare(A,B,Q,R)`

So, can this controller stabilize inverted pendulum for all angles?
So, can this controller stabilize inverted pendulum for all angles?

No!

Linearization error is too large when angle is large
Instead, can we use LQR to track reference trajectory?
Yes

But, first we need to talk about time-varying systems
LQR for Time-Varying Dynamical Systems

\[ x_{t+1} = A_t x_t + B_t u_t \]
LQR for Time-Varying Dynamical Systems

\[ x_{t+1} = A_t x_t + B_t u_t \]

\[ c(x_t, u_t) = x_t^T Q_t x_t + u_t^T R_t u_t \]
LQR for Time-Varying Dynamical Systems

\[ x_{t+1} = A_t x_t + B_t u_t \]

\[ c(x_t, u_t) = x_t^T Q_t x_t + u_t^T R_t u_t \]

Straight forward to get LQR equations

\[ K_t = -(R_t + B_t^T V_{t+1} B_t)^{-1} B_t^T V_{t+1} A_t \]

\[ V_t = Q_t + K_t^T R_t K_t + (A_t + B_t K_t)^T V_{t+1} (A_t + B_t K_t) \]
Why do we care about time-varying?

Ans: Linearization about a trajectory
Linearize about a time-varying trajectory
Linearize about a time-varying trajectory

\[ x_{ref}(t) \quad u_{ref}(t) \]
Linearize about a time-varying trajectory

$x_{ref}(t)$ $u_{ref}(t)$

$x(t)$
Linearize about a time-varying trajectory

\[ \dot{x} = f(x, u) \]

\[ x_{ref}(t) \quad u_{ref}(t) \]

\[ x(t) \]
Linearize about a time-varying trajectory

\[ \dot{x} = f(x, u) \]

\[ A_t = \left. \frac{\partial f}{\partial x} \right|_{x_{\text{ref}}(t)} \]
Linearize about a time-varying trajectory

\[ \dot{x} = f(x, u) \]

\[ A_t = \left. \frac{\partial f}{\partial x} \right|_{x_{\text{ref}}(t)} \]

\[ B_t = \left. \frac{\partial f}{\partial u} \right|_{u_{\text{ref}}(t)} \]
Linearize about a time-varying trajectory

\[
\dot{x} = f(x, u)
\]

\[
A_t = \left. \frac{\partial f}{\partial x} \right|_{x_{ref}(t)}
\]

\[
B_t = \left. \frac{\partial f}{\partial u} \right|_{u_{ref}(t)}
\]

\[
x_{t+1} = A_t x_t + B_t u_t + x_{t}^{off}
\]
Linearize about a time-varying trajectory

\[ \dot{x} = f(x, u) \]

\[ A_t = \left. \frac{\partial f}{\partial x} \right|_{x_{ref}(t)} \]

\[ B_t = \left. \frac{\partial f}{\partial u} \right|_{u_{ref}(t)} \]

\[ x_{t+1} = A_t x_t + B_t u_t + x_{off}^t \]
Making an affine system linear
Making an affine system linear

\[ x_{t+1} = A_t x_t + B_t u_t + x_{t}^{off} \]
Making an affine system linear

\[ x_{t+1} = A_t x_t + B_t u_t + x_t^{off} \]

Homogenous coordinates \( \tilde{x} = \begin{pmatrix} x \\ 1 \end{pmatrix} \)
Making an affine system linear

\[ x_{t+1} = A_t x_t + B_t u_t + x_{t}^{off} \]

Homogenous coordinates

\[ \tilde{x} = \begin{pmatrix} x \\ 1 \end{pmatrix} \]

\[ \tilde{x}_{t+1} = \begin{pmatrix} A_t & x_{t}^{off} \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} B_t \\ 0 \end{pmatrix} u_t \]
Making an affine system linear

\[ x_{t+1} = A_t x_t + B_t u_t + x_{t}^{off} \]

Homogenous coordinates

\[ \tilde{x} = \begin{pmatrix} x \\ 1 \end{pmatrix} \]

\[ \tilde{x}_{t+1} = \begin{pmatrix} A_t & x_{t}^{off} \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} B_t \\ 0 \end{pmatrix} u_t \]

Similarly you can transform cost function

\[ c(\tilde{x}_t, u_t) = \tilde{x}_t^T \tilde{Q}_t \tilde{x}_t + u_t^T R_t u_t \]
Shape of the value function changes along trajectory

\[ x_{ref}(t) \quad u_{ref}(t) \]
Shape of the value function changes along trajectory

\[ x_{ref}(t) \ u_{ref}(t) \]
Shape of the value function changes along trajectory

$x_{ref}(t)$ $u_{ref}(t)$

$x_t$ \quad $x_{t+1}$

$\tilde{A}_t$ \quad \ldots

$\tilde{B}_t$ \quad \ldots

$\tilde{Q}_t$ \quad \ldots

$\tilde{R}_t$ \quad \ldots

$\tilde{K}_t$, $\tilde{V}_t$
Shape of the value function changes along trajectory

\[ x_{ref}(t) \quad u_{ref}(t) \]

\[ x_t \quad x_{t+1} \quad x_{t+2} \]

\[ \tilde{A}_t \quad \tilde{B}_t \quad \tilde{Q}_t \quad \tilde{R}_t \quad \tilde{K}_t, \tilde{V}_t \]
Questions
Questions

1. Can we solve LQR for continuous time dynamics?
Questions

1. Can we solve LQR for continuous time dynamics?

Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)
Questions

1. Can we solve LQR for continuous time dynamics?

   Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?
Questions

1. Can we solve LQR for continuous time dynamics?
   Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?
   Yes! We will talk about iterative LQR next class
Questions

1. Can we solve LQR for continuous time dynamics?

   Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?

   Yes! We will talk about iterative LQR next class

3. What if I want to penalize control derivatives?
Questions

1. Can we solve LQR for continuous time dynamics?

   Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?

   Yes! We will talk about iterative LQR next class

3. What if I want to penalize control derivatives?

   No problem! Add control as part of state space
1. Can we solve LQR for continuous time dynamics?

   Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?

   Yes! We will talk about iterative LQR next class

3. What if I want to penalize control derivatives?

   No problem! Add control as part of state space

4. Can we handle noisy dynamics?
Questions

1. Can we solve LQR for continuous time dynamics?
   Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?
   Yes! We will talk about iterative LQR next class

3. What if I want to penalize control derivatives?
   No problem! Add control as part of state space

4. Can we handle noisy dynamics?
   Yes! Gaussian noise does not change the answer
Trivia: Duality between control and estimation

R. Kalman “A new approach to linear filtering and prediction problems.” (1960)

<table>
<thead>
<tr>
<th>linear-quadratic regulator</th>
<th>Kalman-Bucy filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V$</td>
<td>$\Sigma$</td>
</tr>
<tr>
<td>$A$</td>
<td>$A^T$</td>
</tr>
<tr>
<td>$B$</td>
<td>$H^T$</td>
</tr>
<tr>
<td>$R$</td>
<td>$DD^T$</td>
</tr>
<tr>
<td>$Q$</td>
<td>$CC^T$</td>
</tr>
<tr>
<td>$t$</td>
<td>$t_f - t$</td>
</tr>
</tbody>
</table>

(Table from E. Todorov “General duality between optimal control and estimation”, CDC, 2008)