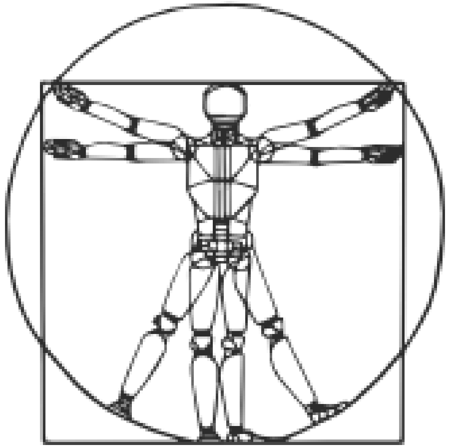
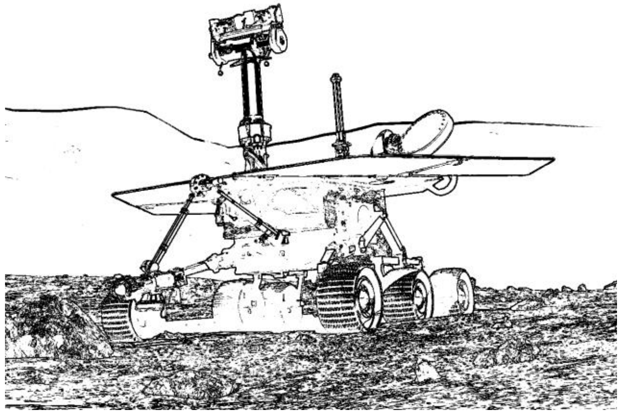




COMP417
Introduction to Robotics and Intelligent Systems

Lecture 7: Linear Quadratic Regulator

Florian Shkurti

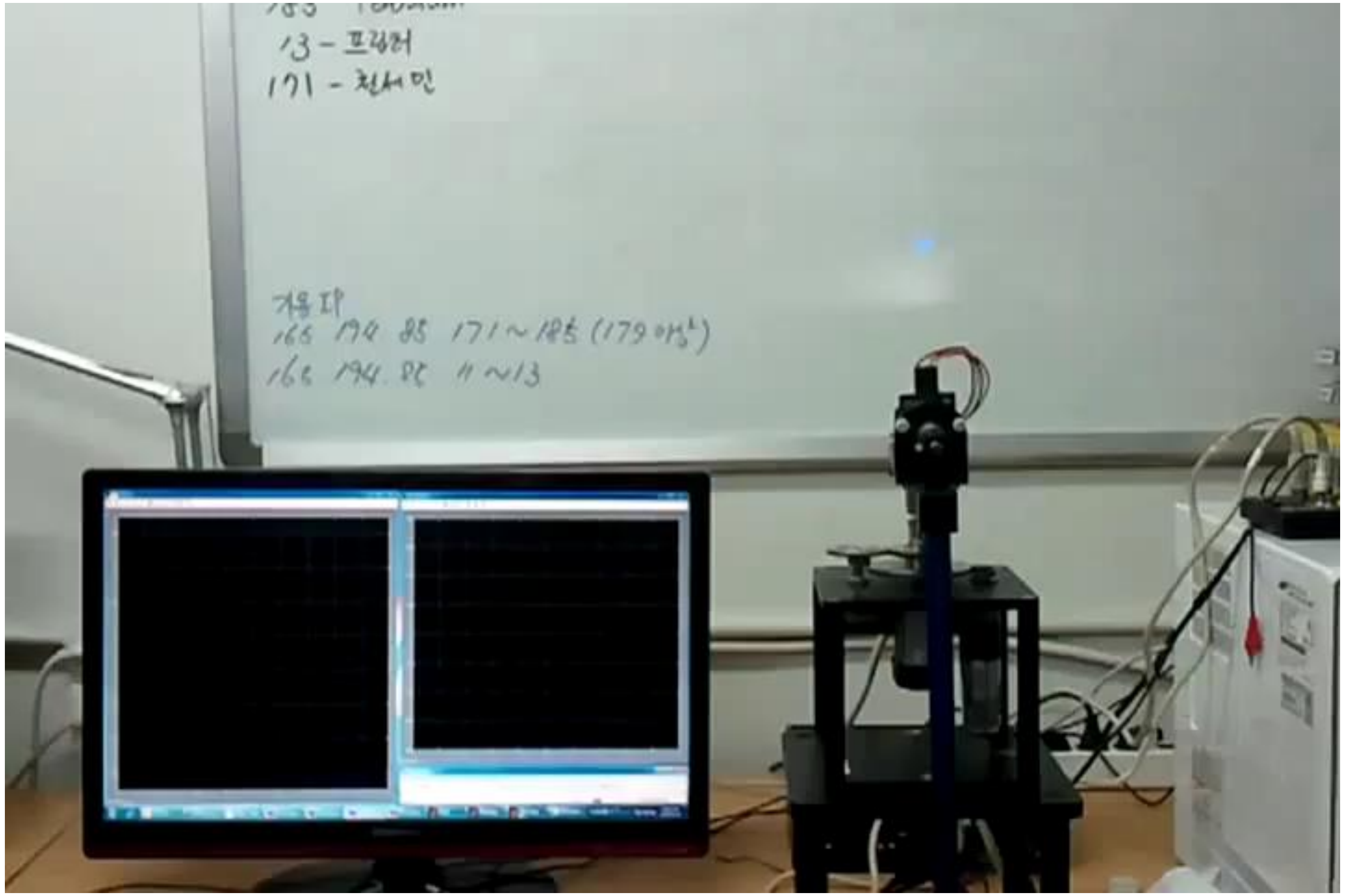
Computer Science Ph.D. student

florian@cim.mcgill.ca

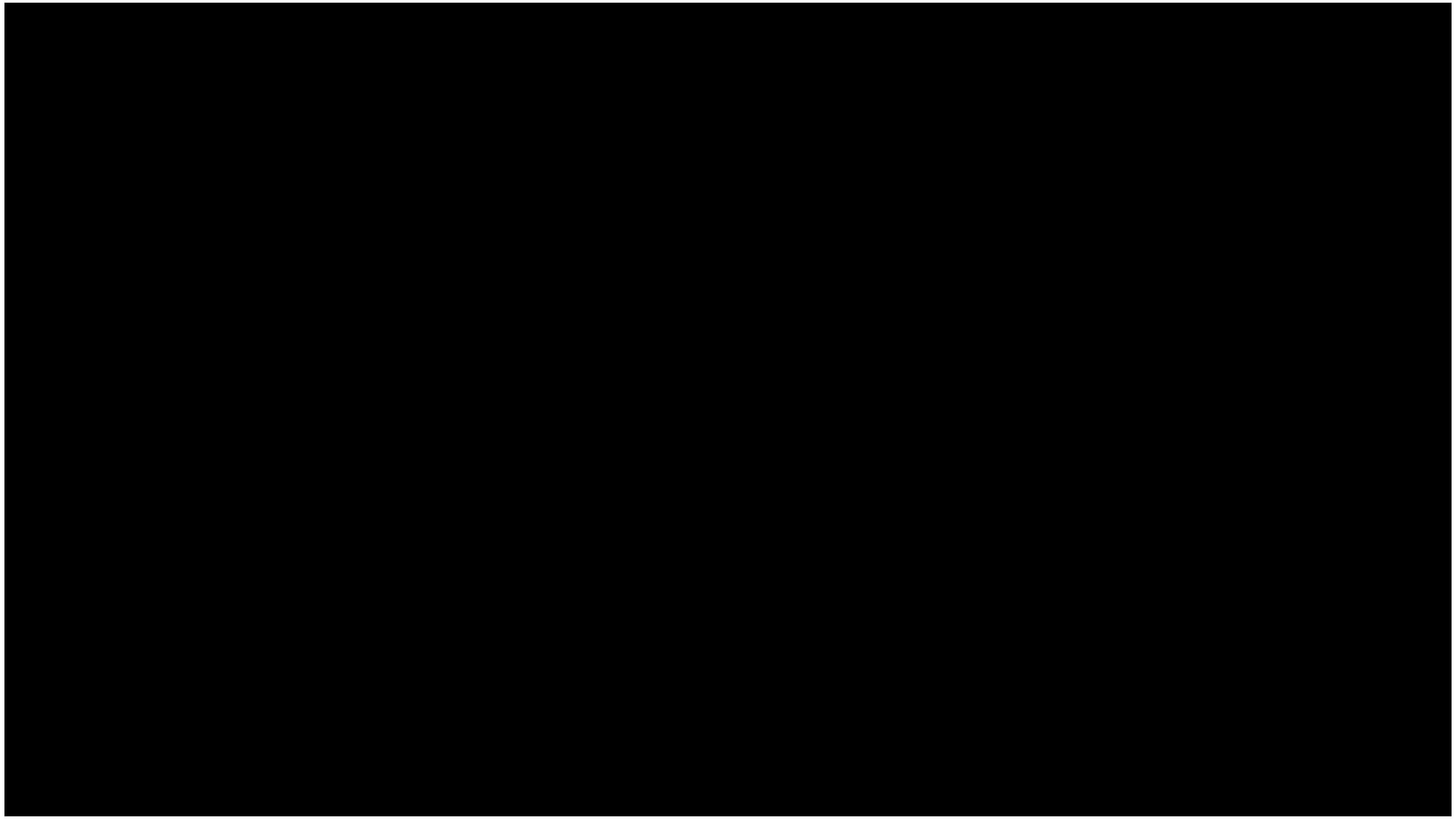
McGill

MRL Mobile Robotics Lab at McGill University

What you can do with LQR controllers



What you can do with LQR controllers



Pieter Abbeel, Helicopter Aerobatics

Why not use PID?

Why not use PID?

- We could, but:

- The gains for PID are good for a small region of state-space.
  - System reaches a state outside this set → becomes unstable
  - PID has no formal guarantees on the size of the set

- We would need to tune PID gains for every control variable.
  - If the state vector has multiple dimensions it becomes harder to tune every control variable in isolation. Need to consider interactions and correlations.

- We would need to tune PID gains for different regions of the state-space and guarantee smooth gain transitions
  - This is called gain scheduling, and it takes a lot of effort and time

Why not use PID?

LQR addresses these problems

- We could, but:
- The gains for PID are good for a small region of state-space.
  - System reaches a state outside this set → becomes unstable
  - PID has no formal guarantees on the size of the set
- We would need to tune PID gains for every control variable.
  - If the state vector has multiple dimensions it becomes harder to tune every control variable in isolation. Need to consider interactions and correlations.
- We would need to tune PID gains for different regions of the state-space and guarantee smooth gain transitions
  - This is called gain scheduling, and it takes a lot of effort and time

LQR: assumptions

- You know the dynamics model of the system
- It is linear: $$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$$

State at the next time step

Control at the next time step

Q: Why is it useful to have a dynamics model of the system?

LQR: assumptions

- You know the dynamics model of the system
- It is linear: $$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$$

State at the next time step

Control at the next time step

Q: Why is it useful to have a dynamics model of the system?
A: Because you can predict the state at a future time step,
given a sequence of control inputs

LQR: assumptions

- You know the dynamics model of the system
- It is linear: $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$

- There is an instantaneous cost associated with being at state $\mathbf{x}_t$ and taking the action $\mathbf{u}_t$: $g(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$

Quadratic state cost: Penalizes deviation from the zero vector

Quadratic control cost: Penalizes high control signals

LQR: assumptions

- You know the dynamics model of the system
- It is linear: $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$

- There is an instantaneous cost associated with being at state $\mathbf{x}_t$ and taking the action $\mathbf{u}_t$: $g(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$

Square matrices Q and R must be positive definite:

$$Q = Q^T \quad \text{and} \quad \forall x, x^T Q x > 0$$
$$R = R^T \quad \text{and} \quad \forall u, u^T R u > 0$$

i.e. positive cost for ANY nonzero state or control vector

LQR: assumptions

- You know the dynamics model of the system
- It is linear: $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$

- There is an instantaneous cost associated with being at state $\mathbf{x}_t$ and taking the action $\mathbf{u}_t$: $g(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$

Square matrices Q and R must be positive definite:

$$Q = Q^T \quad \text{and} \quad \forall x, x^T Q x > 0$$
$$R = R^T \quad \text{and} \quad \forall u, u^T R u > 0$$

i.e. positive cost for ANY nonzero state or control vector

Note: Recall that a positive definite matrix has positive eigenvalues

LQR: assumptions

- You know the dynamics model of the system
- It is linear: $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$

- There is an instantaneous cost associated with being at state $\mathbf{x}_t$ and taking the action $\mathbf{u}_t$: $g(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q\mathbf{x}_t + \mathbf{u}_t^T R\mathbf{u}_t$

- You either know the state $\mathbf{x}_t$ directly or you can estimate it

LQR: goal

- Stabilize the system around state $\mathbf{x}_t = \mathbf{0}$ with control $\mathbf{u}_t = \mathbf{0}$
- Then $\mathbf{x}_{t+1} = \mathbf{0}$ and the system will remain at zero.

Which systems are linear?

- Omnidirectional robot

$$
\begin{aligned}
x_{t+1} &= x_t + v_x(t)\delta t \\
y_{t+1} &= y_t + v_y(t)\delta t \\
\theta_{t+1} &= \theta_t + \omega_z(t)\delta t
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
\mathbf{x}_{t+1} &= I\mathbf{x}_t + \delta t I \mathbf{u}_t \\
\mathbf{x}_{t+1} &= A\mathbf{x}_t + B\mathbf{u}_t
\end{aligned}
$$

- Simple car?

$$
\begin{aligned}
x_{t+1} &= x_t + v_x(t)\cos(\theta_t)\delta t \\
y_{t+1} &= y_t + v_x(t)\sin(\theta_t)\delta t \\
\theta_{t+1} &= \theta_t + \omega_z\delta t
\end{aligned}
\quad \Rightarrow \quad
\mathbf{x}_{t+1} = I\mathbf{x}_t + \begin{bmatrix} \delta t\cos(\theta_t) & 0 & 0 \\ 0 & \delta t\sin(\theta_t) & 0 \\ 0 & 0 & \delta t \end{bmatrix} \mathbf{u}_t
$$

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B(\mathbf{x}_t)\mathbf{u}_t$$

Which systems are linear?

- Omnidirectional robot

$$\begin{aligned} x_{t+1} &= x_t + v_x(t)\delta t \\ y_{t+1} &= y_t + v_y(t)\delta t \\ \theta_{t+1} &= \theta_t + \omega_z(t)\delta t \end{aligned} \quad \Rightarrow \quad \begin{aligned} \mathbf{x}_{t+1} &= I\mathbf{x}_t + \delta t I\mathbf{u}_t \\ \mathbf{x}_{t+1} &= A\mathbf{x}_t + B\mathbf{u}_t \end{aligned}$$

- Simple car? NO

$$\begin{aligned} x_{t+1} &= x_t + v_x(t)\text{cos}(\theta_t)\delta t \\ y_{t+1} &= y_t + v_x(t)\text{sin}(\theta_t)\delta t \\ \theta_{t+1} &= \theta_t + \omega_z\delta t \end{aligned} \quad \Rightarrow \quad \mathbf{x}_{t+1} = I\mathbf{x}_t + \begin{bmatrix} \delta t\text{cos}(\theta_t) & 0 & 0 \\ 0 & \delta t\text{sin}(\theta_t) & 0 \\ 0 & 0 & \delta t \end{bmatrix} \mathbf{u}_t$$

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B(\mathbf{x}_t)\mathbf{u}_t$$

Which systems are linear?

- Linearity means if you scale or add to the input the output will also reflect the scaling and addition

- Formally, the dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ are linear if

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$$

Finite-Horizon LQR

- Idea: finding controls is an optimization problem
- Compute the control variables that minimize the cumulative cost

$$
\begin{aligned}
\underset{u_0,\ldots,u_N}{\operatorname{argmin}} & \quad \sum_{t=0}^{t=N} g(\mathbf{x}_t, \mathbf{u}_t) \\
\text{s.t.} & \\
\mathbf{x}_1 & = A\mathbf{x}_0 + B\mathbf{u}_0 \\
\mathbf{x}_2 & = A\mathbf{x}_1 + B\mathbf{u}_1 \\
& \ldots \\
\mathbf{x}_N & = A\mathbf{x}_{N-1} + B\mathbf{u}_{N-1}
\end{aligned}
$$

Finite-Horizon LQR

- Idea: finding controls is an optimization problem
- Compute the control variables that minimize the cumulative cost

$$
\begin{aligned}
\underset{u_0,\ldots,u_N}{\operatorname{argmin}} \quad & \sum_{t=0}^{t=N} g(\mathbf{x}_t, \mathbf{u}_t) \\
\text{s.t.} \quad & \\
\mathbf{x}_1 &= A\mathbf{x}_0 + B\mathbf{u}_0 \\
\mathbf{x}_2 &= A\mathbf{x}_1 + B\mathbf{u}_1 \\
& \ldots \\
\mathbf{x}_N &= A\mathbf{x}_{N-1} + B\mathbf{u}_{N-1}
\end{aligned}
$$

We could solve this as a constrained nonlinear optimization problem. But, there is a better way: we can find a closed-form solution.

Finding the LQR controller in closed-form by recursion

- Let $J_n(\mathbf{x})$ denote the cumulative cost-to-go starting from state x and moving for n time steps.

- I.e. cumulative future cost from now till n more steps

- $J_0(\mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$ is the terminal cost of ending up at state x, with no actions left to do. Let's denote it $J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$

Q: What is the optimal cumulative cost-to-go function with 1 time step left?

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$J_1(\mathbf{x}) = \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})]$$

Bellman update (a.k.a. Dynamic Programming)

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned} J_1(\mathbf{x}) &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})] \\ &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \end{aligned}$$

Q: How do we optimize a multivariable function with respect to some variables (in our case, the controls)?

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned}
J_1(\mathbf{x}) &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})] \\
&= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})]
\end{aligned}$$

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned}
J_1(\mathbf{x}) &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})] \\
&= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + 2\mathbf{u}^T B^T P_0 A \mathbf{x} + \mathbf{u}^T B^T P_0 B \mathbf{u}]
\end{aligned}$$

Q: How do we optimize a multivariable function with respect to some variables (in our case, the controls)?

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned}
J_1(\mathbf{x}) &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})] \\
&= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + \min_{\mathbf{u}} [\underbrace{\mathbf{u}^T R \mathbf{u}}_{\text{Quadratic term in u}} + \underbrace{2\mathbf{u}^T B^T P_0 A \mathbf{x}}_{\text{Linear term in u}} + \underbrace{\mathbf{u}^T B^T P_0 B \mathbf{u}}_{\text{Quadratic term in u}}]
\end{aligned}$$

A: Take the partial derivative w.r.t. controls and set it to zero. That will give you a critical point.

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q\mathbf{x} + \mathbf{x}^T A^T P_0 A\mathbf{x} + \min_{\mathbf{u}}[\mathbf{u}^T R\mathbf{u} + 2\mathbf{u}^T B^T P_0 A\mathbf{x} + \mathbf{u}^T B^T P_0 B\mathbf{u}]$$

From calculus/algebra:

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M\mathbf{u}) = (M + M^T)\mathbf{u}$$

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M\mathbf{b}) = M\mathbf{b}$$

If M is symmetric:

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M\mathbf{u}) = 2M\mathbf{u}$$

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q\mathbf{x} + \mathbf{x}^T A^T P_0 A\mathbf{x} + \min_{\mathbf{u}}[\mathbf{u}^T R\mathbf{u} + 2\mathbf{u}^T B^T P_0 A\mathbf{x} + \mathbf{u}^T B^T P_0 B\mathbf{u}]$$

The minimum is attained at:

$$2R\mathbf{u} + 2B^T P_0 A\mathbf{x} + 2B^T P_0 B\mathbf{u} = \mathbf{0}$$

$$(R + B^T P_0 B)\mathbf{u} = -B^T P_0 A\mathbf{x}$$

Q: Is this matrix invertible? Recall R, Po are positive definite matrices.

From calculus/algebra:

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M\mathbf{u}) = (M + M^T)\mathbf{u}$$

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M\mathbf{b}) = M\mathbf{b}$$

If M is symmetric:

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M\mathbf{u}) = 2M\mathbf{u}$$

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q\mathbf{x} + \mathbf{x}^T A^T P_0 A\mathbf{x} + \min_{\mathbf{u}}[\mathbf{u}^T R\mathbf{u} + 2\mathbf{u}^T B^T P_0 A\mathbf{x} + \mathbf{u}^T B^T P_0 B\mathbf{u}]$$

The minimum is attained at:

$$2R\mathbf{u} + 2B^T P_0 A\mathbf{x} + 2B^T P_0 B\mathbf{u} = \mathbf{0}$$

$$(R + B^T P_0 B)\mathbf{u} = -B^T P_0 A\mathbf{x}$$

Q: Is this matrix invertible? Recall R, Po are positive definite matrices.

$R + B^T P_0 B$ is positive definite, so it is invertible

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + 2\mathbf{u}^T B^T P_0 A \mathbf{x} + \mathbf{u}^T B^T P_0 B \mathbf{u}]$$

The minimum is attained at:

$$2R\mathbf{u} + 2B^T P_0 A\mathbf{x} + 2B^T P_0 B\mathbf{u} = \mathbf{0}$$

$$(R + B^T P_0 B)\mathbf{u} = -B^T P_0 A\mathbf{x}$$

So, the optimal control for the last time step is:

$$\mathbf{u} = -(R + B^T P_0 B)^{-1} B^T P_0 A\mathbf{x}$$

$$\mathbf{u} = K_1 \mathbf{x}$$

Linear controller in terms of the state

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q\mathbf{x} + \mathbf{x}^T A^T P_0 A\mathbf{x} + \min_{\mathbf{u}}[\mathbf{u}^T R\mathbf{u} + 2\mathbf{u}^T B^T P_0 A\mathbf{x} + \mathbf{u}^T B^T P_0 B\mathbf{u}]$$

The minimum is attained at:

$$2R\mathbf{u} + 2B^T P_0 A\mathbf{x} + 2B^T P_0 B\mathbf{u} = \mathbf{0}$$

$$(R + B^T P_0 B)\mathbf{u} = -B^T P_0 A\mathbf{x}$$

So, the optimal control for the last time step is:

$$\mathbf{u} = -(R + B^T P_0 B)^{-1} B^T P_0 A\mathbf{x}$$

$$\mathbf{u} = K_1\mathbf{x}$$

Linear controller in terms of the state

We computed the location of the minimum.
Now, plug it back in and compute the
minimum value

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned}
J_1(\mathbf{x}) &= \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + 2\mathbf{u}^T B^T P_0 A \mathbf{x} + \mathbf{u}^T B^T P_0 B \mathbf{u}] \\
&= \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + [\mathbf{x}^T K_1^T R K_1 \mathbf{x} + 2\mathbf{x}^T K_1^T B^T P_0 A \mathbf{x} + \mathbf{x}^T K_1^T B^T P_0 B K_1 \mathbf{x}] \\
&= \mathbf{x}^T (Q + K_1^T R K_1 + (A + BK_1)^T P_0 (A + BK_1)) \mathbf{x}
\end{aligned}$$

Q: Why is this a big deal?
A: The cost-to-go function remains quadratic after the first recursive step.

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\mathbf{u} = -(R + B^T P_0 B)^{-1} B^T P_0 A \mathbf{x}$$
$$\mathbf{u} = K_1 \mathbf{x}$$

$$\begin{aligned} J_1(\mathbf{x}) &= \mathbf{x}^T (Q + K_1^T R K_1 + (A + BK_1)^T P_0 (A + BK_1)) \mathbf{x} \\ &= \mathbf{x}^T P_1 \mathbf{x} \end{aligned}$$

...

In fact the recursive steps generalize

$$\mathbf{u} = -(R + B^T P_{n-1} B)^{-1} B^T P_{n-1} A \mathbf{x}$$
$$\mathbf{u} = K_n \mathbf{x}$$

$$\begin{aligned} J_n(\mathbf{x}) &= \mathbf{x}^T (Q + K_n^T R K_n + (A + BK_n)^T P_{n-1} (A + BK_n)) \mathbf{x} \\ &= \mathbf{x}^T P_n \mathbf{x} \end{aligned}$$

Finite-Horizon LQR: algorithm summary

$$P_0 = Q$$

// n is the # of steps left

for n = 1…N

$$K_n = -(R + B^T P_{n-1} B)^{-1} B^T P_{n-1} A$$

$$P_n = Q + K_n^T R K_n + (A + BK_n)^T P_{n-1} (A + BK_n)$$

Optimal controller for i-step horizon is $\mathbf{u}(\mathbf{x}) = K_i \mathbf{x}$ with cost-to-go $J_i(\mathbf{x}) = \mathbf{x}^T P_i \mathbf{x}$

Finite-Horizon LQR: algorithm summary

$$P_0 \quad = \quad Q$$

// n is the # of steps left

for n = 1…N

$$K_n = -(R + B^T P_{n-1} B)^{-1} B^T P_{n-1} A$$

$$P_n \quad = \quad Q + K_n^T R K_n + (A + BK_n)^T P_{n-1} (A + BK_n)$$

Matrix gains are precomputed based on the dynamics model and the instantaneous cost

Then you execute the policy, issuing commands based on state feedback

Optimal controller for i-step horizon is $\mathbf{u}(\mathbf{x}) = K_i \mathbf{x}$ with cost-to-go $J_i(\mathbf{x}) = \mathbf{x}^T P_i \mathbf{x}$

A state-dependent controller is called a policy

Finite-Horizon LQR: algorithm summary

Potential problem for states of dimension >> 100:
Matrix inversion is expensive: O(n^2.3) for the best known algorithm and O(n^3) for Gaussian Elimination.

$$P_0 = Q$$

// n is the # of steps left

for n = 1…N

$$K_n = -(R + B^T P_{n-1} B)^{-1} B^T P_{n-1} A$$

$$P_n = Q + K_n^T R K_n + (A + BK_n)^T P_{n-1} (A + BK_n)$$

Optimal controller for i-step horizon is $\mathbf{u}(\mathbf{x}) = K_i \mathbf{x}$ with cost-to-go $J_i(\mathbf{x}) = \mathbf{x}^T P_i \mathbf{x}$

LQR extensions: nonlinear dynamics

What can we do when $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$?

We want to stabilize the system around state $\mathbf{x}_t = \mathbf{0}$
But with nonlinear dynamics we do not know if $\mathbf{u}_t = \mathbf{0}$ will keep the system at the zero state.

LQR extensions: nonlinear dynamics

What can we do when $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$?

We want to stabilize the system around state $\mathbf{x}_t = \mathbf{0}$
But with nonlinear dynamics we do not know if $\mathbf{u}_t = \mathbf{0}$ will keep the system at the zero state.
→ Need to compute $\mathbf{u}^*$ such that $\mathbf{0}_{t+1} = f(\mathbf{0}_t, \mathbf{u}^*)$

LQR extensions: nonlinear dynamics

What can we do when $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$?

We want to stabilize the system around state $\mathbf{x}_t = \mathbf{0}$
But with nonlinear dynamics we do not know if $\mathbf{u}_t = \mathbf{0}$ will keep the system at the zero state.
→ Need to compute $\mathbf{u}^*$ such that $\mathbf{0}_{t+1} = f(\mathbf{0}_t, \mathbf{u}^*)$

Taylor expansion: linearize the nonlinear dynamics around the point $(\mathbf{0}, \mathbf{u}^*)$

$$\mathbf{x}_{t+1} \simeq f(\mathbf{0}, \mathbf{u}^*) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{0}, \mathbf{u}^*)(\mathbf{x}_t - \mathbf{0}) + \frac{\partial f}{\partial \mathbf{u}}(\mathbf{0}, \mathbf{u}^*)(\mathbf{u}_t - \mathbf{u}^*)$$

$$\mathbf{x}_{t+1} \simeq A\mathbf{x}_t + B(\mathbf{u}_t - \mathbf{u}^*)$$

LQR extensions: nonlinear dynamics

What can we do when $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$?

We want to stabilize the system around state $\mathbf{x}^* = \mathbf{0}$
But with nonlinear dynamics we do not know if $\mathbf{u}^* = \mathbf{0}$ will keep the system at the zero state.
→ Need to compute $\mathbf{u}^*$ such that $\mathbf{0}_{t+1} = f(\mathbf{0}_t, \mathbf{u}^*)$

Taylor expansion: linearize the nonlinear dynamics around the point $(\mathbf{0}, \mathbf{u}^*)$

$$\mathbf{x}_{t+1} \simeq f(\mathbf{0}, \mathbf{u}^*) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{0}, \mathbf{u}^*)(\mathbf{x}_t - \mathbf{0}) + \frac{\partial f}{\partial \mathbf{u}}(\mathbf{0}, \mathbf{u}^*)(\mathbf{u}_t - \mathbf{u}^*)$$

$$\mathbf{x}_{t+1} \simeq A\mathbf{x}_t + B(\mathbf{u}_t - \mathbf{u}^*)$$

Create new variable $\bar{\mathbf{u}}_t = \mathbf{u}_t - \mathbf{u}^*$ and apply LQR iterations to the linear system $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\bar{\mathbf{u}}_t$

LQR extensions: nonlinear dynamics

What can we do when $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$?

We want to stabilize the system around state $\mathbf{x}^* = \mathbf{0}$
But with nonlinear dynamics we do not know if $\mathbf{u}^* = \mathbf{0}$ will keep the system at the zero state.
→ Need to compute $\mathbf{u}^*$ such that $\mathbf{0}_{t+1} = f(\mathbf{0}_t, \mathbf{u}^*)$

Taylor expansion: linearize the nonlinear dynamics around the point $(\mathbf{0}, \mathbf{u}^*)$

$$\mathbf{x}_{t+1} \simeq f(\mathbf{0}, \mathbf{u}^*) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{0}, \mathbf{u}^*)(\mathbf{x}_t - \mathbf{0}) + \frac{\partial f}{\partial \mathbf{u}}(\mathbf{0}, \mathbf{u}^*)(\mathbf{u}_t - \mathbf{u}^*)$$

$$\mathbf{x}_{t+1} \simeq A\mathbf{x}_t + B(\mathbf{u}_t - \mathbf{u}^*)$$

Create new variable $\bar{\mathbf{u}}_t = \mathbf{u}_t - \mathbf{u}^*$ and apply LQR iterations to the

Unfortunately with the linearization you pay a price: You have to provide an initial state close to the zero state, otherwise you might not get good results

LQR extensions: non-quadratic cost

What can we do when $g(\mathbf{x}_t, \mathbf{u}_t)$ is not quadratic?

Same trick as before: use Taylor expansion to get the quadratic terms, but matrices involved are not guaranteed to be positive-definite, so this doesn't always work.

LQR extensions: time-varying systems

- What can we do when $\mathbf{x}_{t+1} = A_t \mathbf{x}_t + B_t \mathbf{u}_t$ and $g(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$?
- Turns out, the proof and the algorithm are almost the same

$$P_0 \quad = \quad Q_N$$

// n is the # of steps left

for n = 1…N

$$K_n = -(R_{N-n} + B_{N-n}^T P_{n-1} B_{N-n})^{-1} B_{N-n}^T P_{n-1} A_{N-n}$$

$$P_n \quad = \quad Q_{N-n} + K_n^T R_{N-n} K_n + (A_{N-n} + B_{N-n} K_n)^T P_{n-1} (A_{N-n} + B_{N-n} K_n)$$

Optimal controller for i-step horizon is $\mathbf{u}(\mathbf{x}) = K_i \mathbf{x}$ with cost-to-go $J_i(\mathbf{x}) = \mathbf{x}^T P_i \mathbf{x}$

LQR extensions: trajectory following

- You are given a reference trajectory (not just path, but states and times, or states and controls) that needs to be approximated

$$\mathbf{x}_0^*, \mathbf{x}_1^*, \ldots, \mathbf{x}_N^* \qquad \mathbf{u}_0^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_N^*$$

Linearize the nonlinear dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ around the reference point $(\mathbf{x}_t^*, \mathbf{u}_t^*)$

$$\mathbf{x}_{t+1} \simeq f(\mathbf{x}_t^*, \mathbf{u}_t^*) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}_t^*, \mathbf{u}_t^*)(\mathbf{x}_t - \mathbf{x}_t^*) + \frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}_t^*, \mathbf{u}_t^*)(\mathbf{u}_t - \mathbf{u}_t^*)$$

$$\begin{aligned} \bar{\mathbf{x}}_{t+1} &\simeq A_t \bar{\mathbf{x}}_t + B_t \bar{\mathbf{u}}_t \\ g(\mathbf{x}_t, \mathbf{u}_t) &= \bar{\mathbf{x}}_t^T Q \bar{\mathbf{x}}_t + \bar{\mathbf{u}}_t^T R \bar{\mathbf{u}}_t \end{aligned} \quad \text{where} \quad \begin{aligned} \bar{\mathbf{x}}_t &= \mathbf{x}_t - \mathbf{x}_t^* \\ \bar{\mathbf{u}}_t &= \mathbf{u}_t - \mathbf{u}_t^* \end{aligned}$$

Trajectory following can be implemented as a time-varying LQR approximation. Not always clear if this is the best way though.

LQR example #1: omnidirectional vehicle with friction

- Similar to double integrator dynamical system, but with friction:

$$\underbrace{m\ddot{\mathbf{p}}}_{\substack{\text{Force applied}\\ \text{to the vehicle}}} = \underbrace{\mathbf{u}}_{\substack{\text{Control applied}\\ \text{to the vehicle}}} - \underbrace{\alpha\dot{\mathbf{p}}}_{\substack{\text{Friction opposed to}\\ \text{motion}}}$$

LQR example #1: omnidirectional vehicle with friction

- Similar to double integrator dynamical system, but with friction:

$$m\ddot{\mathbf{p}} = \mathbf{u} - \alpha\dot{\mathbf{p}}$$

- Set $\dot{\mathbf{p}} = \mathbf{v}$ and then you get:

$$m\dot{\mathbf{v}} = \mathbf{u} - \alpha\mathbf{v}$$

LQR example #1: omnidirectional vehicle with friction

- Similar to double integrator dynamical system, but with friction:

$$m\ddot{\mathbf{p}} = \mathbf{u} - \alpha\dot{\mathbf{p}}$$

- Set $\dot{\mathbf{p}} = \mathbf{v}$ and then you get:

$$m\dot{\mathbf{v}} = \mathbf{u} - \alpha\mathbf{v}$$

- We discretize by setting

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad\qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha\mathbf{v}_t$$

LQR example #1: omnidirectional vehicle with friction

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad\qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha \mathbf{v}_t$$

- Define the state vector $\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$

Q: How can we express this as a linear system?

LQR example #1: omnidirectional vehicle with friction

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad\qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha \mathbf{v}_t$$

- Define the state vector $\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t + \frac{\delta t}{m}\mathbf{u}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix} \mathbf{u}_t$$

LQR example #1: omnidirectional vehicle with friction

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad\qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha \mathbf{v}_t$$

- Define the state vector $\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t + \frac{\delta t}{m}\mathbf{u}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 - \alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1 - \alpha\delta t/m \end{bmatrix} \mathbf{x}_t + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix} \mathbf{u}_t$$

LQR example #1: omnidirectional vehicle with friction

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha \mathbf{v}_t$$

- Define the state vector $\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t + \frac{\delta t}{m}\mathbf{u}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1-\alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1-\alpha\delta t/m \end{bmatrix}}_{\mathbf{A}} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}_t$$

LQR example #1: omnidirectional vehicle with friction

- Define the state vector $\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t + \frac{\delta t}{m}\mathbf{u}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} = \overset{\textbf{A}}{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 - \alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1 - \alpha\delta t/m \end{bmatrix}} \mathbf{x}_t + \overset{\textbf{B}}{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}} \mathbf{u}_t$$

- Define the instantaneous cost function

$$\begin{aligned} g(\mathbf{x}, \mathbf{u}) &= \mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} \\ &= \mathbf{x}^T \mathbf{x} + \rho \mathbf{u}^T \mathbf{u} \\ &= ||\mathbf{x}||^2 + \rho ||\mathbf{u}||^2 \end{aligned}$$

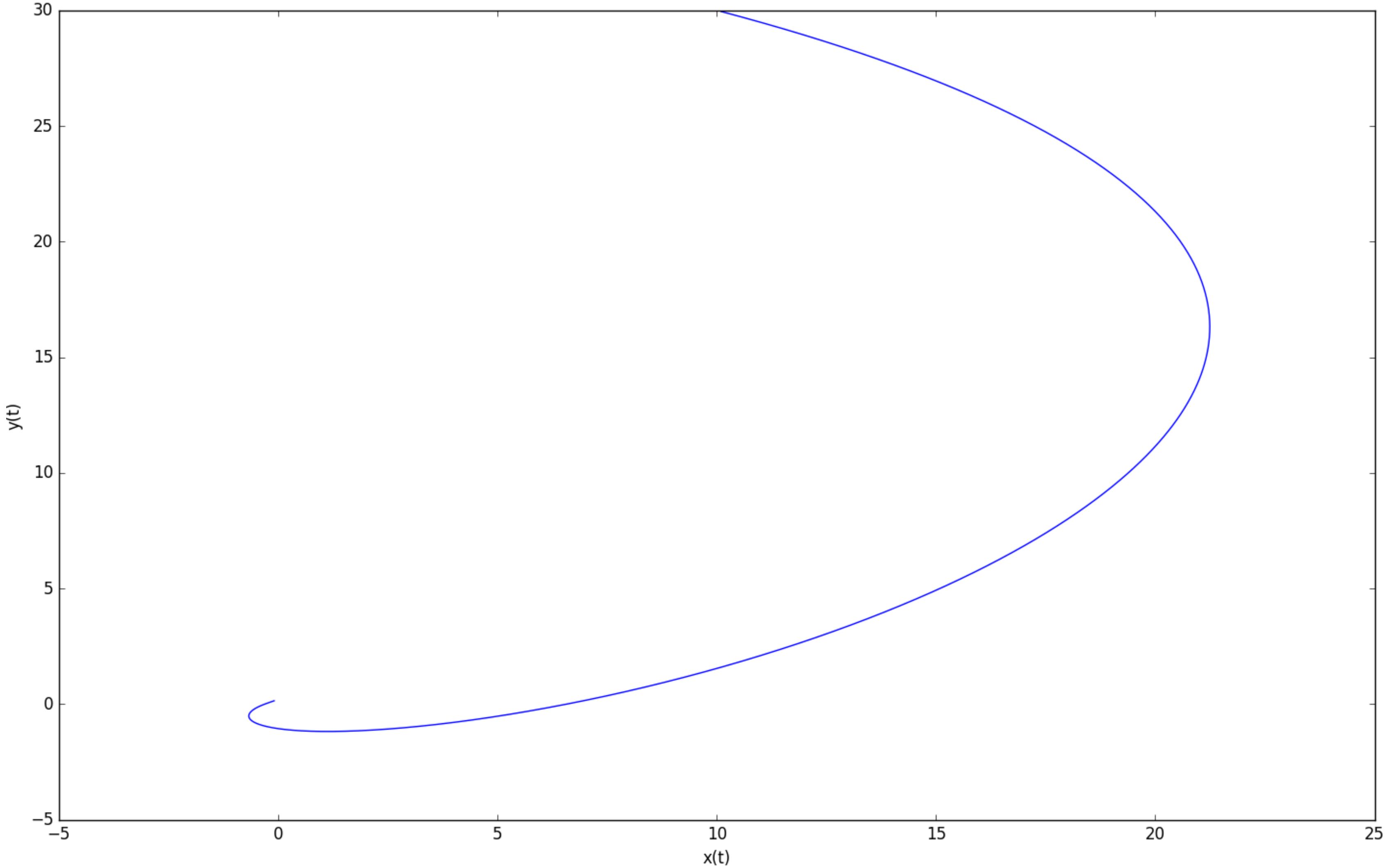
LQR example #1: omnidirectional vehicle with friction

With initial state

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$g(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



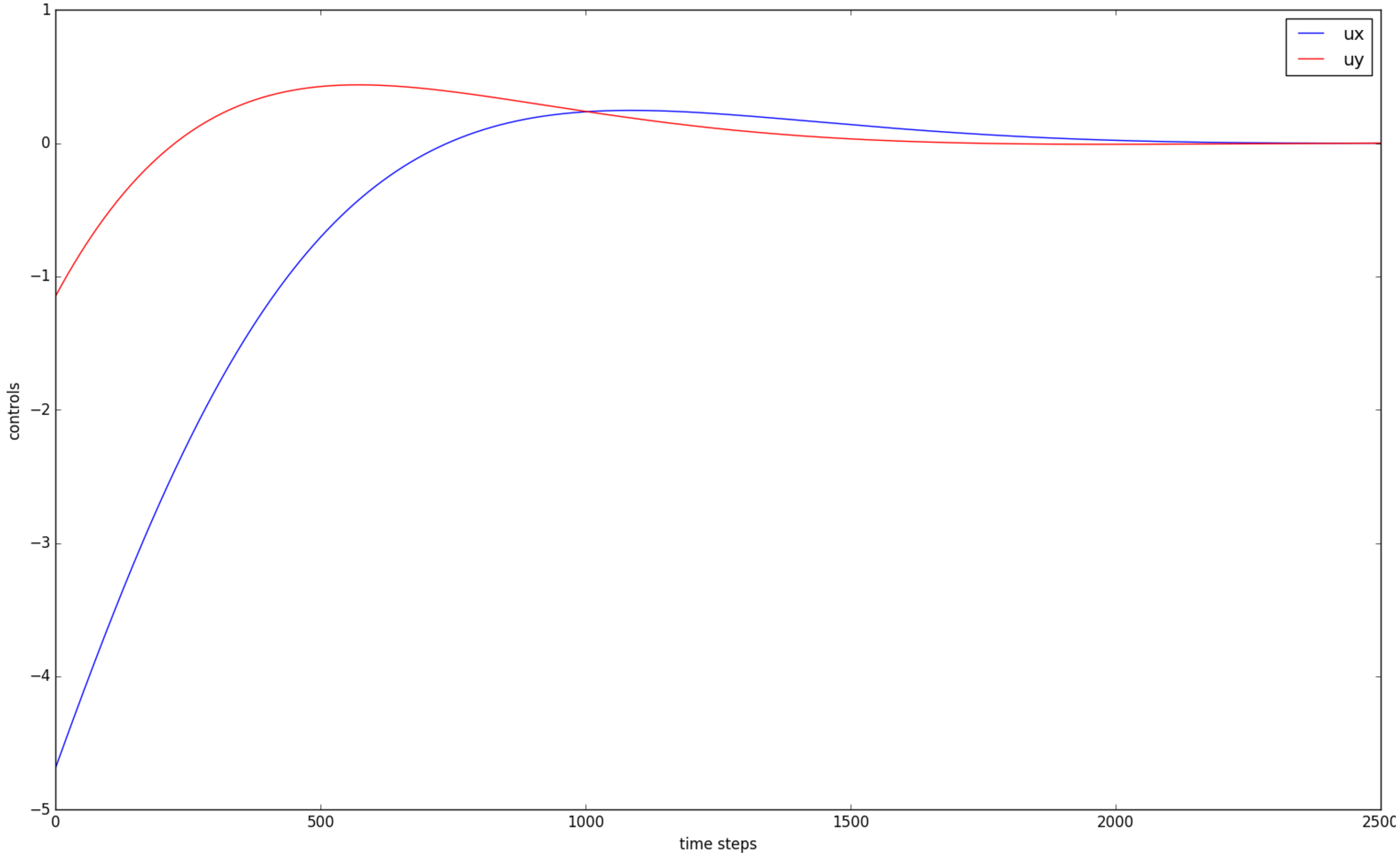
LQR example #1: omnidirectional vehicle with friction

With initial state

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$g(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



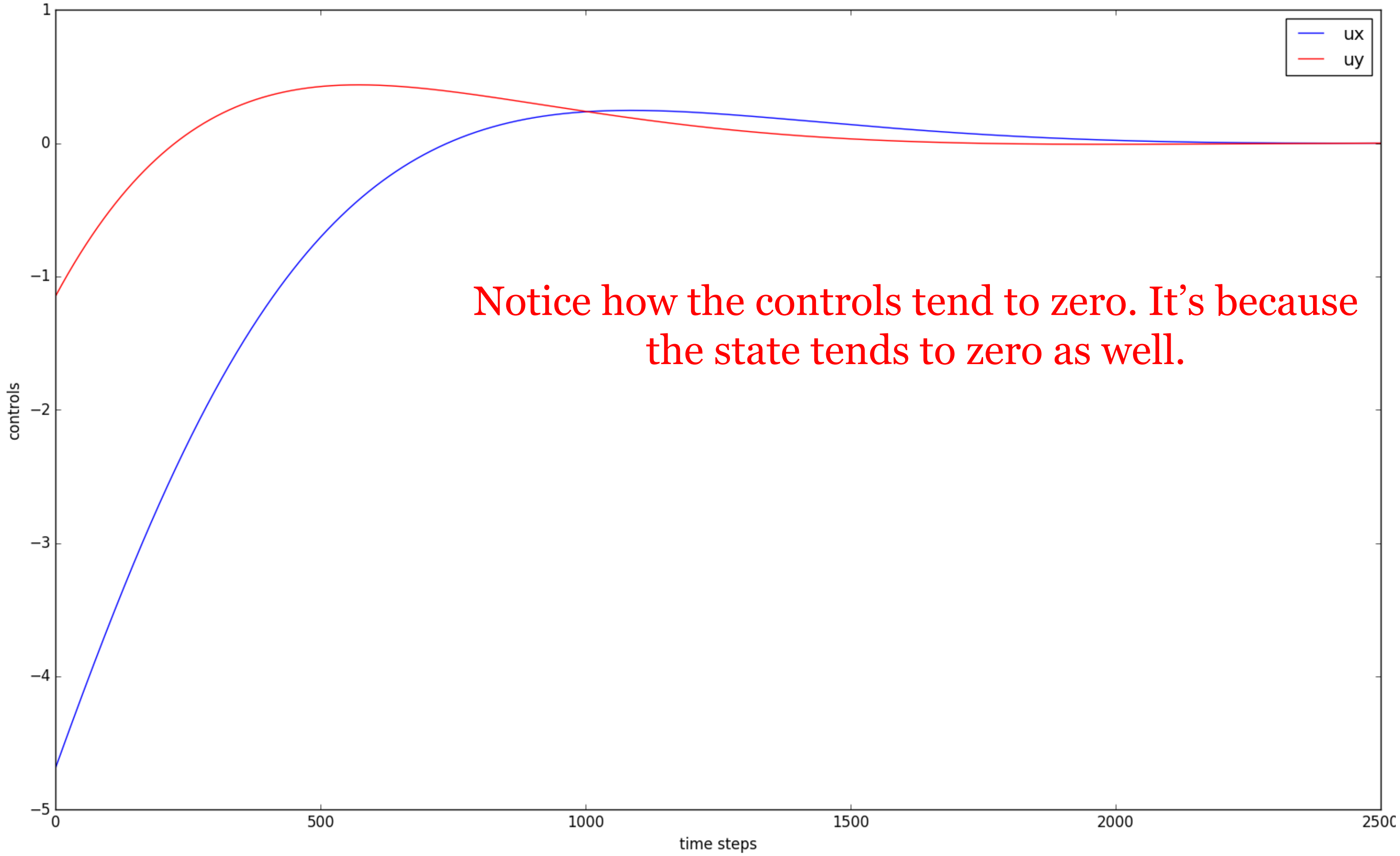
LQR example #1: omnidirectional vehicle with friction

With initial state

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$g(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



Notice how the controls tend to zero. It's because the state tends to zero as well.

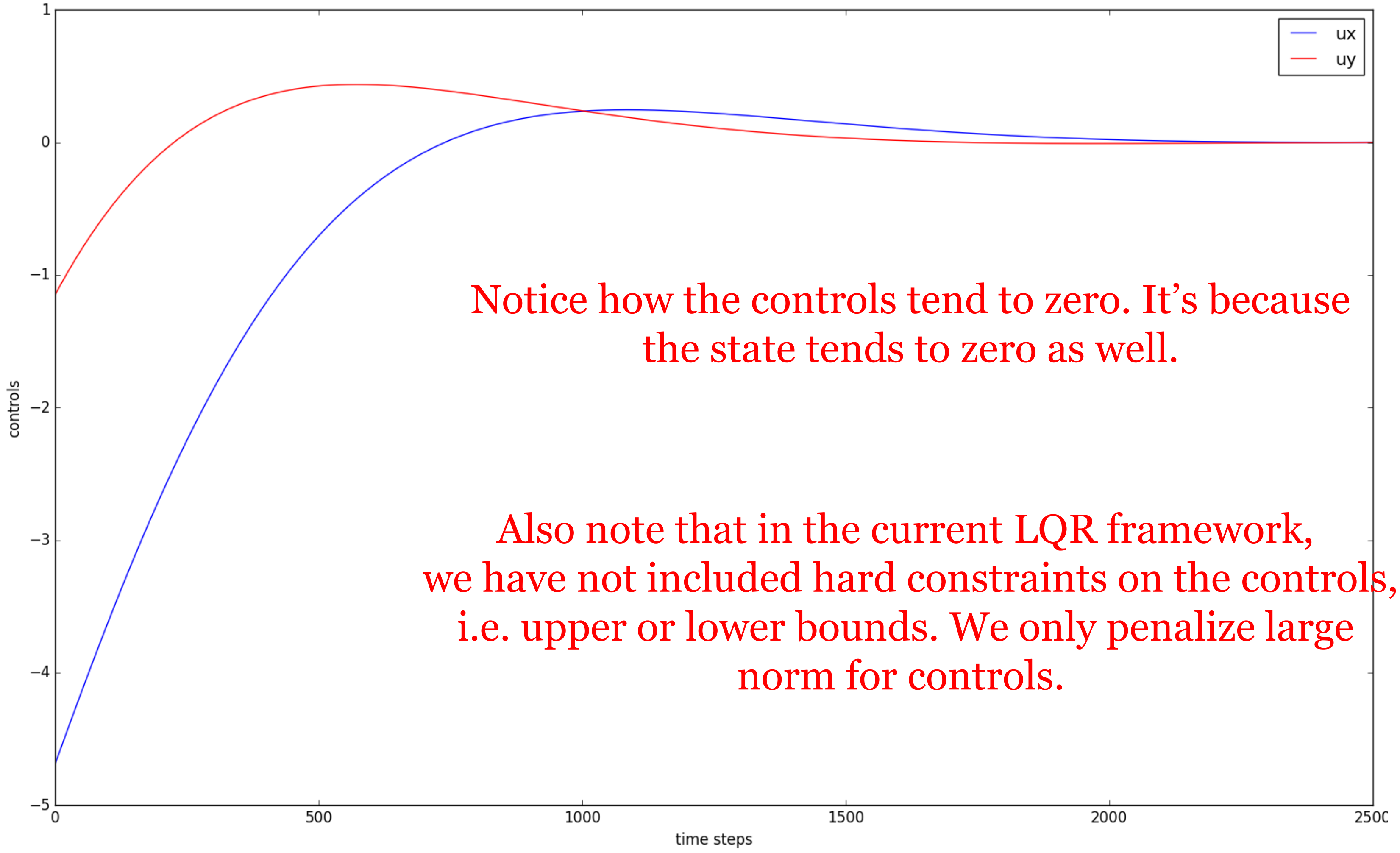
LQR example #1: omnidirectional vehicle with friction

With initial state

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$g(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



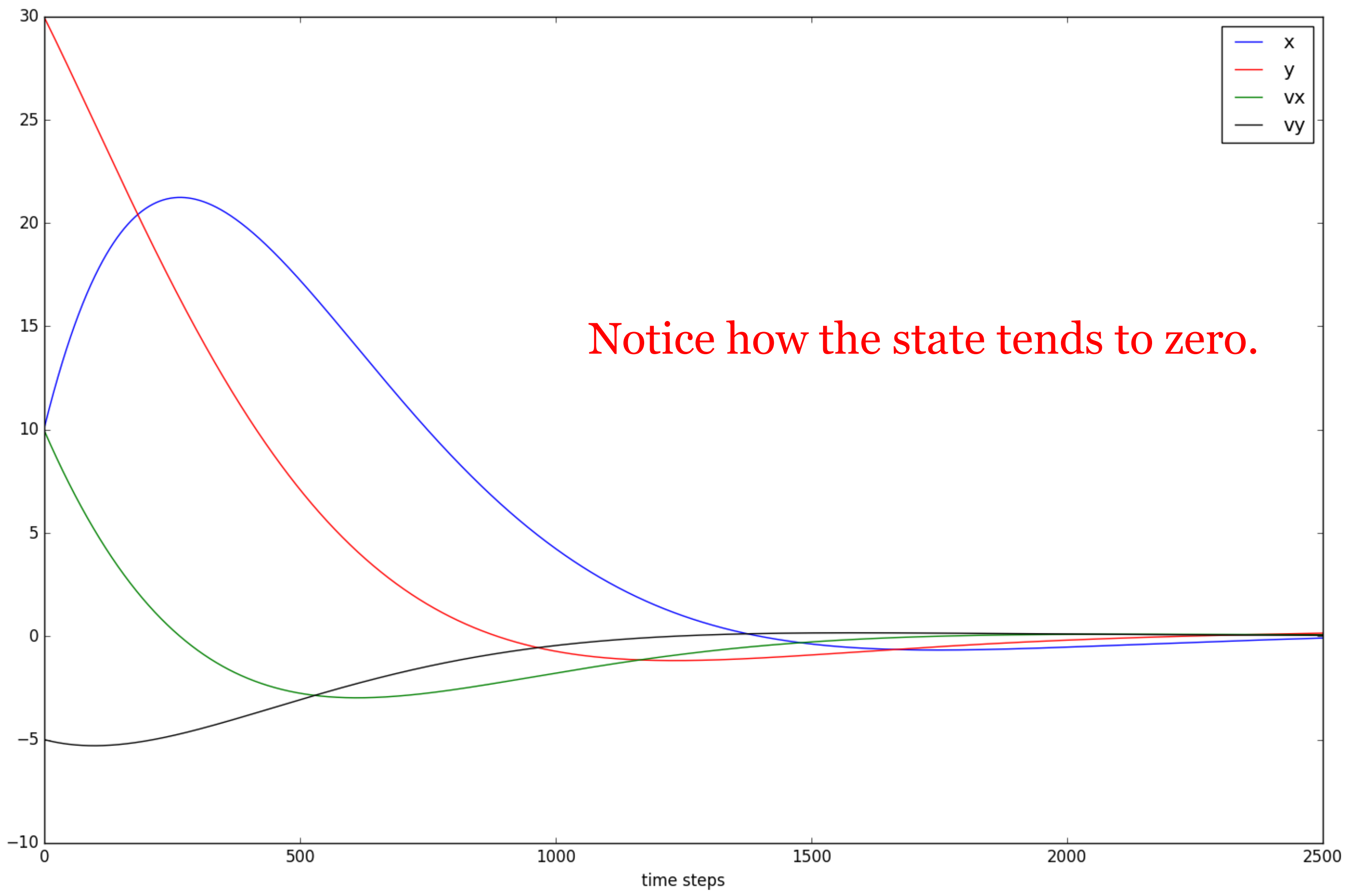
LQR example #1: omnidirectional vehicle with friction

With initial state

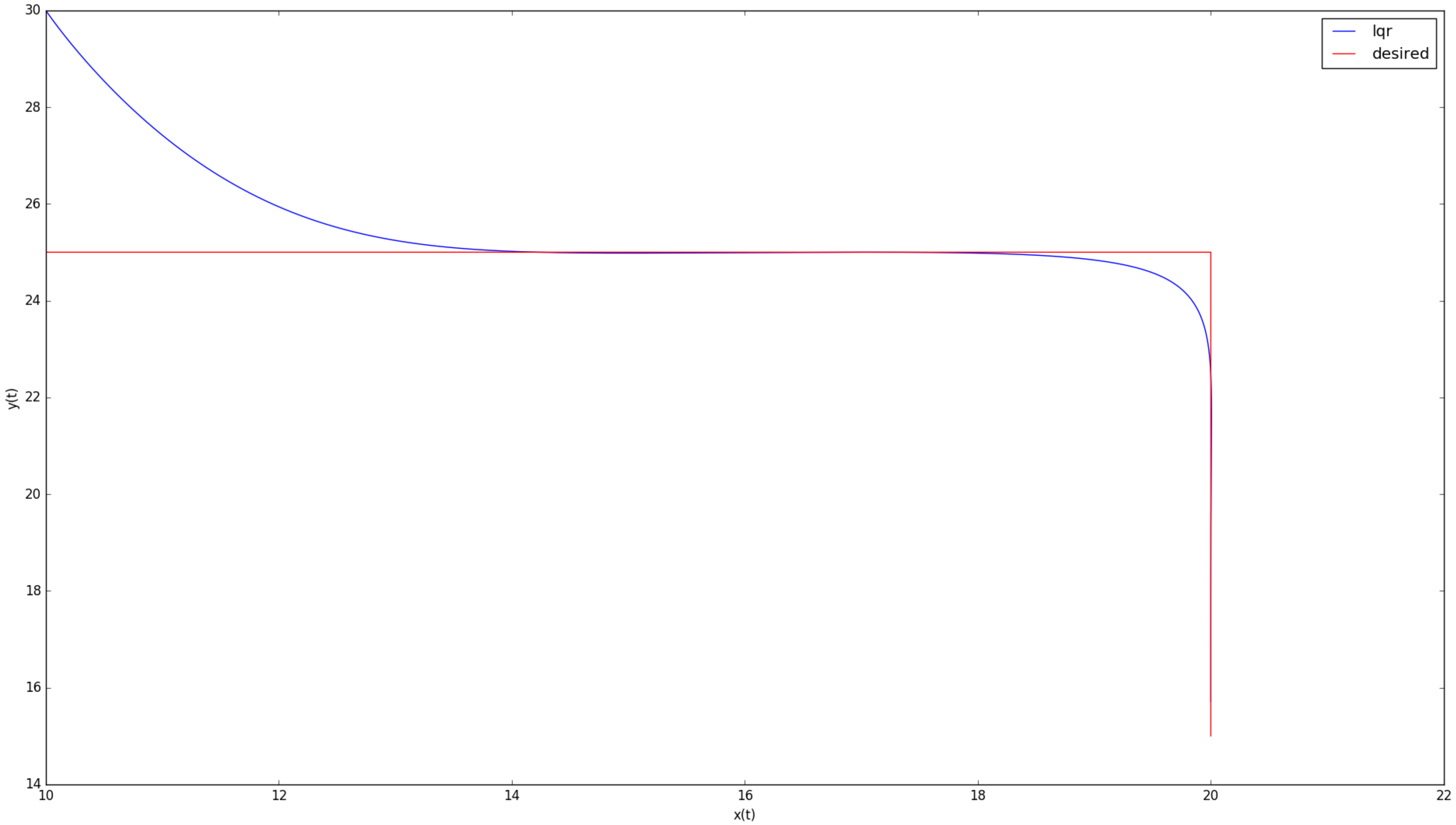
$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$g(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



LQR example #2: trajectory following for omnidirectional vehicle



LQR example #2: trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 - \alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1 - \alpha\delta t/m \end{bmatrix}}_{\mathbf{A}} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}_t$$

We are given a desired trajectory $\mathbf{p}_0^*, \mathbf{p}_1^*, \ldots, \mathbf{p}_T^*$

Instantaneous cost $g(\mathbf{x}_t, \mathbf{u}_t) = (\mathbf{p}_t - \mathbf{p}_t^*)^T Q (\mathbf{p}_t - \mathbf{p}_t^*) + \mathbf{u}_t^T R \mathbf{u}_t$

LQR example #2:
trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1-\alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1-\alpha\delta t/m \end{bmatrix}}_{\mathbf{A}} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}_t$$

Define

$$\begin{aligned} \bar{\mathbf{x}}_{t+1} &= \mathbf{x}_{t+1} - \mathbf{x}^*_{t+1} \\ &= A\mathbf{x}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} \\ &= A\bar{\mathbf{x}}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} + A\mathbf{x}^*_t \end{aligned}$$

(We want $\bar{\mathbf{x}}_{t+1} = \bar{A}\bar{\mathbf{x}}_t + \bar{B}\mathbf{u}_t$)

LQR example #2: trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1-\alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1-\alpha\delta t/m \end{bmatrix}}_{\mathbf{A}} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}_t$$

$$\begin{aligned} \text{Define} \quad \bar{\mathbf{x}}_{t+1} &= \mathbf{x}_{t+1} - \mathbf{x}^*_{t+1} \\ &= A\mathbf{x}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} \\ &= A\bar{\mathbf{x}}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} + A\mathbf{x}^*_t \end{aligned}$$

(We want $\bar{\mathbf{x}}_{t+1} = \bar{A}\bar{\mathbf{x}}_t + \bar{B}\mathbf{u}_t$)

Need to get rid of this additive term

LQR example #2: trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1-\alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1-\alpha\delta t/m \end{bmatrix}}_{\mathbf{A}} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}_t$$

Define
$$\begin{aligned} \bar{\mathbf{x}}_{t+1} &= \mathbf{x}_{t+1} - \mathbf{x}^*_{t+1} \\ &= A\mathbf{x}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} \\ &= A\bar{\mathbf{x}}_t + B\mathbf{u}_t \underbrace{- \mathbf{x}^*_{t+1} + A\mathbf{x}^*_t}_{\mathbf{c}} \end{aligned}$$

(We want $\bar{\mathbf{x}}_{t+1} = \bar{A}\bar{\mathbf{x}}_t + \bar{B}\mathbf{u}_t$)

Need to get rid of this additive term

Redefine state: $\mathbf{z}_{t+1} = \begin{bmatrix} \bar{\mathbf{x}}_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A & c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_t \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \mathbf{u}_t = \bar{A}\mathbf{z}_t + \bar{B}\mathbf{u}_t$

LQR example #2: trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1-\alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1-\alpha\delta t/m \end{bmatrix}}_{\mathbf{A}} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}_t$$

Define
$$\begin{aligned} \bar{\mathbf{x}}_{t+1} &= \mathbf{x}_{t+1} - \mathbf{x}^*_{t+1} \\ &= A\mathbf{x}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} \\ &= A\bar{\mathbf{x}}_t + B\mathbf{u}_t \underbrace{- \mathbf{x}^*_{t+1} + A\mathbf{x}^*_t}_{\mathbf{c}} \end{aligned}$$

(We want $\bar{\mathbf{x}}_{t+1} = \bar{A}\bar{\mathbf{x}}_t + \bar{B}\mathbf{u}_t$)

Need to get rid of this additive term

Redefine state: $\mathbf{z}_{t+1} = \begin{bmatrix} \bar{\mathbf{x}}_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A & c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_t \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \mathbf{u}_t = \bar{A}\mathbf{z}_t + \bar{B}\mathbf{u}_t$

Redefine cost function: $g(\mathbf{z}_t, \mathbf{u}_t) = \mathbf{z}_t^T \bar{Q} \mathbf{z}_t + \mathbf{u}_t^T R \mathbf{u}_t$

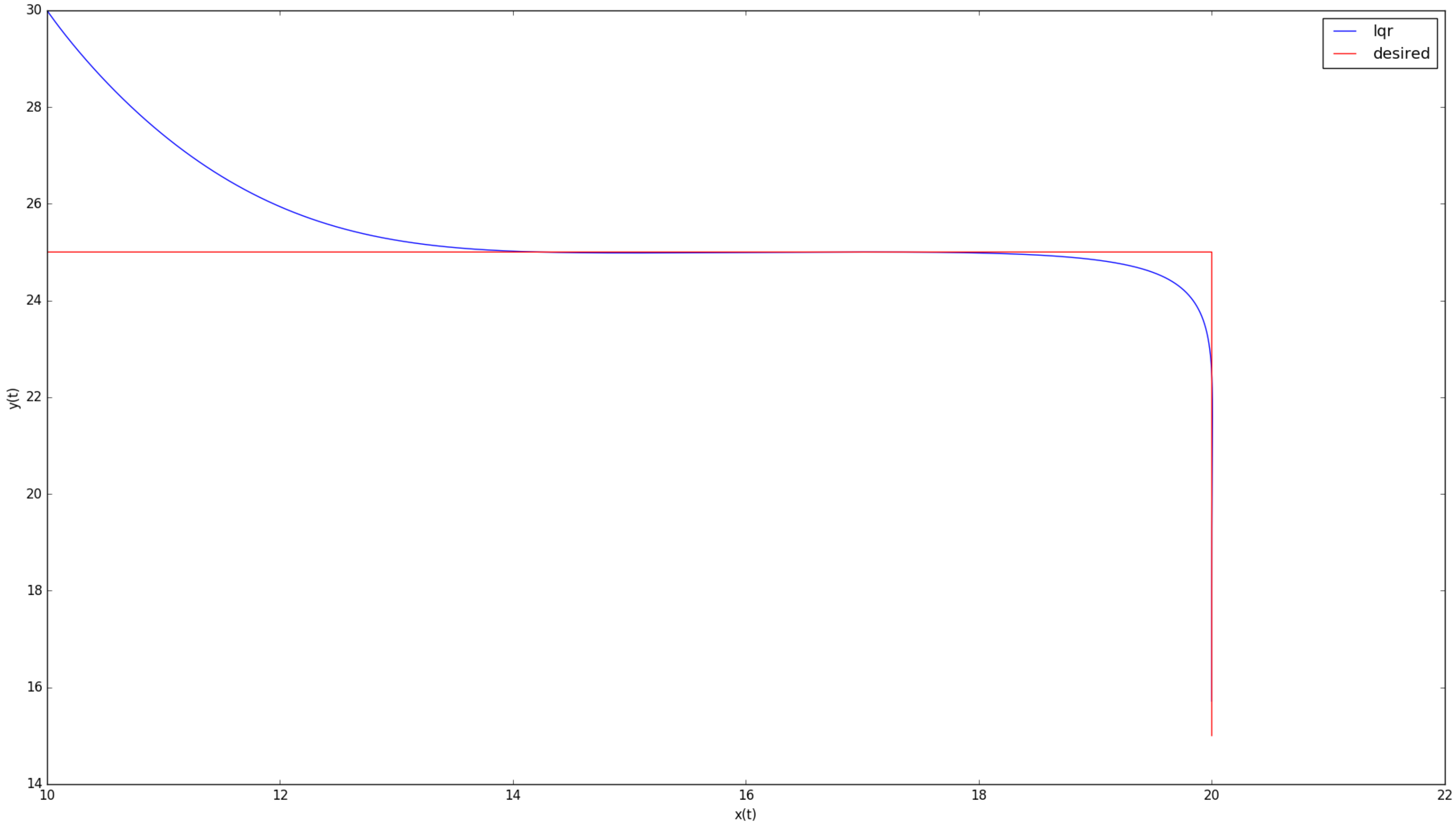
LQR example #2: trajectory following for omnidirectional vehicle

With initial state

$$\mathbf{z}_0 = \begin{bmatrix} 10 \\ 30 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Instantaneous cost function

$$g(\mathbf{z}, \mathbf{u}) = ||\mathbf{z}||^2 + ||\mathbf{u}||^2$$



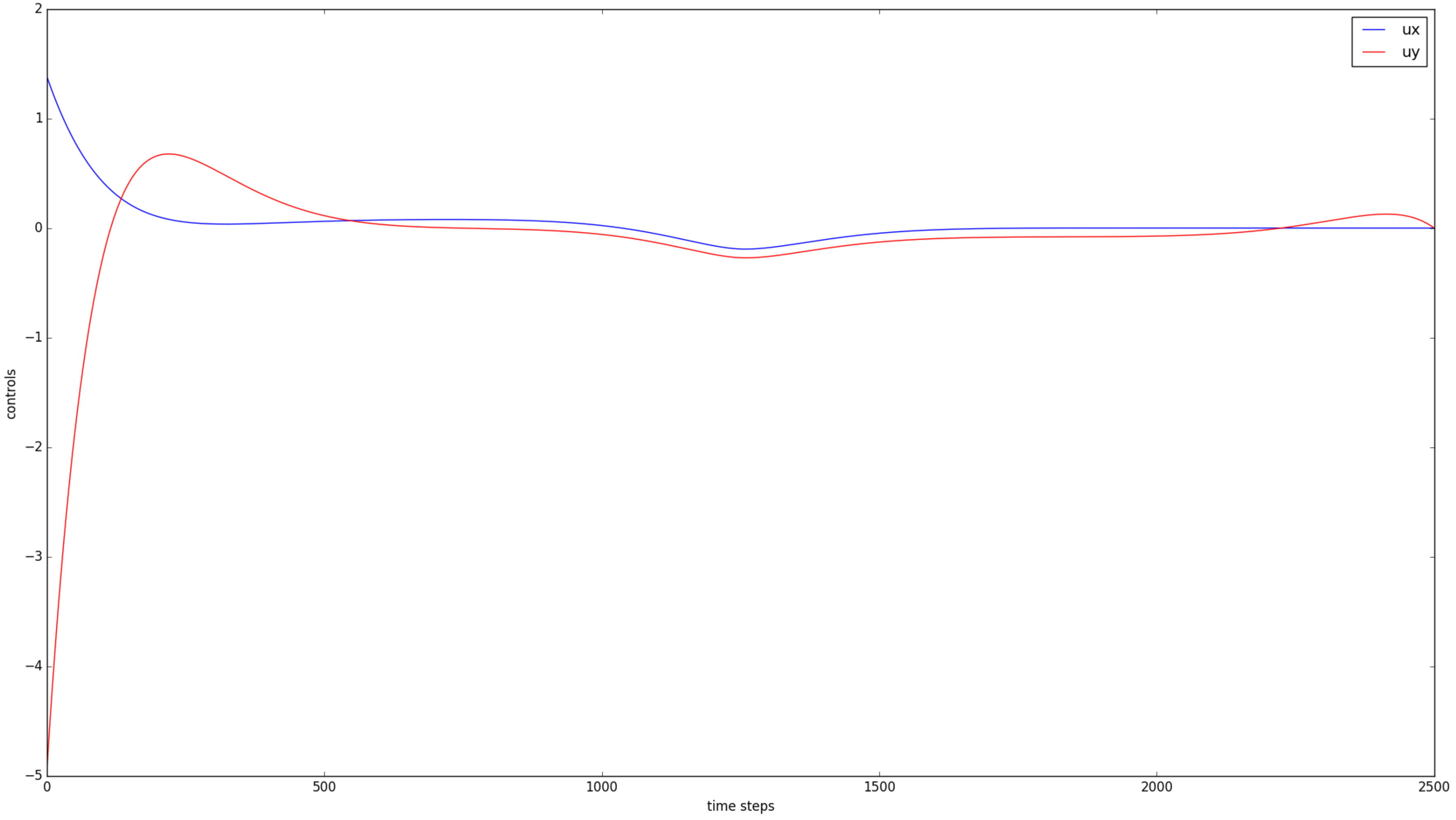
LQR example #2: trajectory following for omnidirectional vehicle

With initial state

$$\mathbf{z}_0 = \begin{bmatrix} 10 \\ 30 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Instantaneous cost function

$$g(\mathbf{z}, \mathbf{u}) = ||\mathbf{z}||^2 + ||\mathbf{u}||^2$$



LQR examples: code to replicate these results

- https://github.com/florianshkurti/comp417.git
- Look under comp417/lqr_examples/python

LQR summary

- Advantages:
  - If system is linear LQR gives the optimal controller that takes the system from state A to state B

- Drawbacks:
  - Need to specify final time at which goal should be reached
  - What if you don’t allow enough time to reach the goal?
  - How can you include obstacles or constraints in the specification?