ME 234(b): Optimal Control

Anushri Dixit

Spring 2022

Slides adapted from CMS 159 (by U. Rosolia) and Berkeley ME231 (by F. Borrelli, M. Morari, C. Jones)

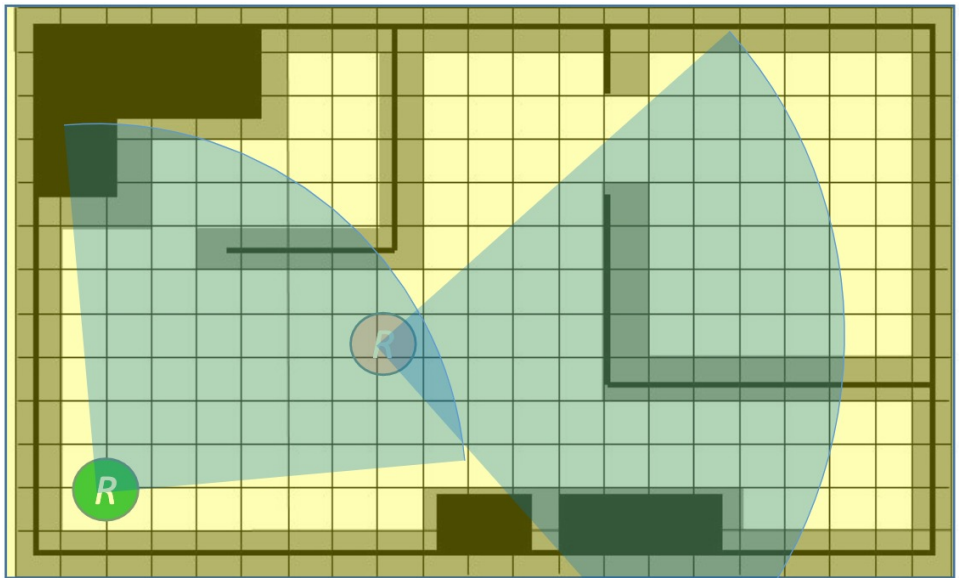
Caltech

Motivation

So far, we have looked at *discrete state and action spaces*.

Suppose we have a set of discrete states to visit and desired discrete actions (say through MDP-based planning).

How does a robot plan continuous actions to visit these states while also accounting for its own dynamics?



Caltech

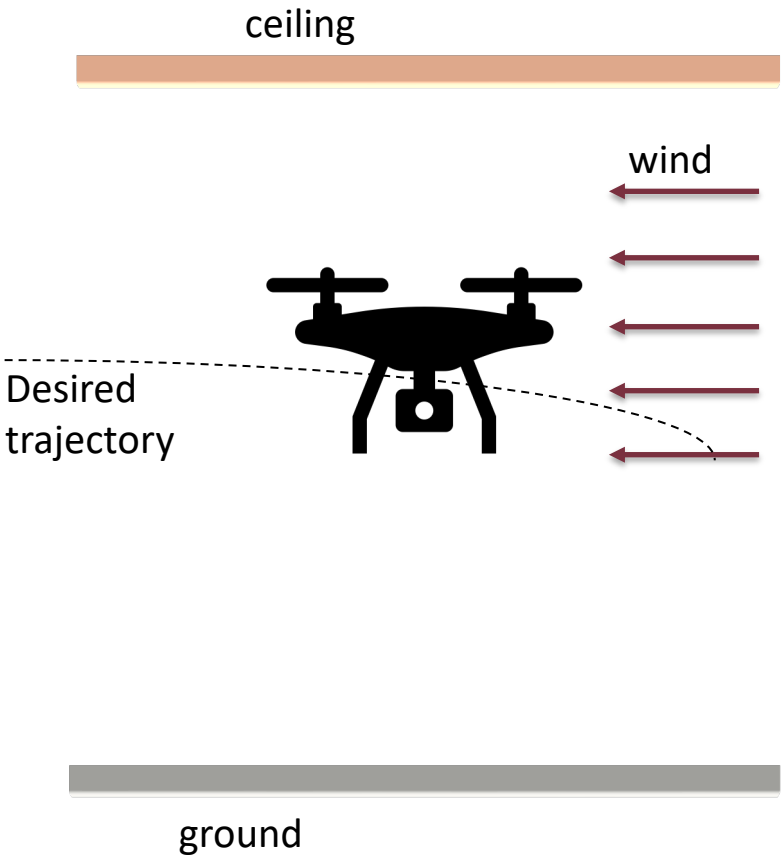
Motivation

Consider a drone in hover.

We want it to:

1. Track a desired trajectory,
2. While not crashing into the ceiling or the ground,
3. And account for disturbances.

In today's lecture we will try to solve 1 without worrying about 2 and 3.



Caltech

Problem formulation: Drone example

Consider a drone governed by the dynamics :

$$\ddot{x} = -g\theta,\ \ddot{y} = g\phi,\ \ddot{z} = g - \frac{u_1}{m}$$

$$\ddot{\phi} = \frac{u_2}{I_{xx}},\ \ddot{\theta} = \frac{u_3}{I_{yy}},\ \ddot{\psi} = \frac{u_4}{I_{zz}}$$

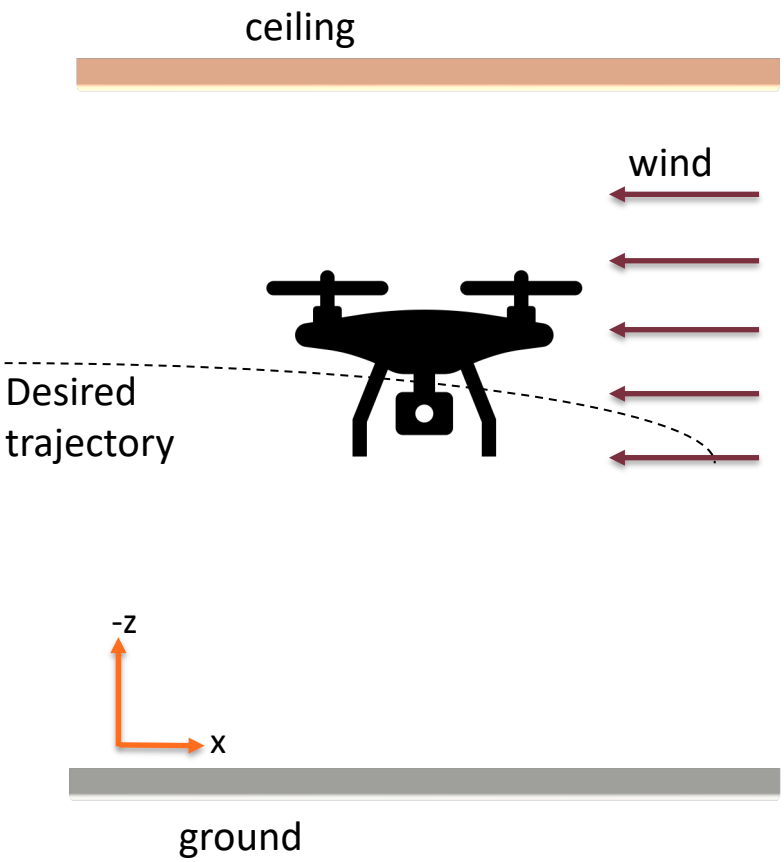
State vector:

$$\boldsymbol{x} = \begin{bmatrix} z & \dot{z} & \psi & \dot{\psi} & x & \dot{x} & \phi & \dot{\phi} & y & \dot{y} & \theta & \dot{\theta} \end{bmatrix}^T$$

$$\boldsymbol{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T$$

Linear dynamics:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$



P. Wang, Z. Man, Z. Cao, J. Zheng and Y. Zhao, "Dynamics modelling and linear control of quadcopter," *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2016, pp. 498-503, doi: 10.1109/ICAMechS.2016.7813499.

Caltech

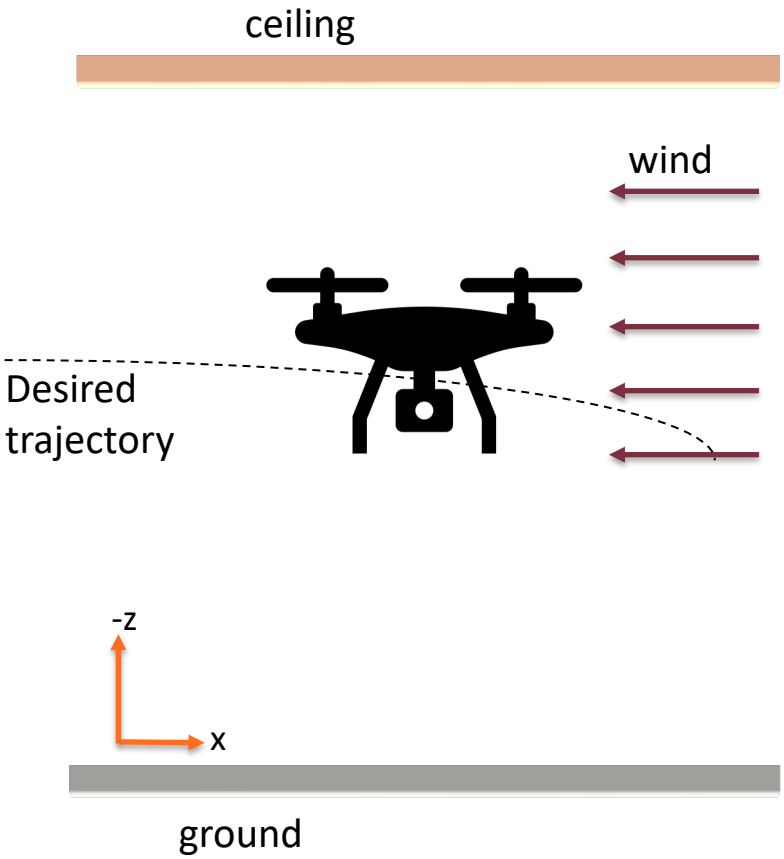
Problem formulation: Drone example

Linear dynamics:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix}$$



P. Wang, Z. Man, Z. Cao, J. Zheng and Y. Zhao, "Dynamics modelling and linear control of quadcopter," *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2016, pp. 498-503, doi: 10.1109/ICAMechS.2016.7813499.

Caltech

Problem formulation: Drone example

Linear dynamics:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$

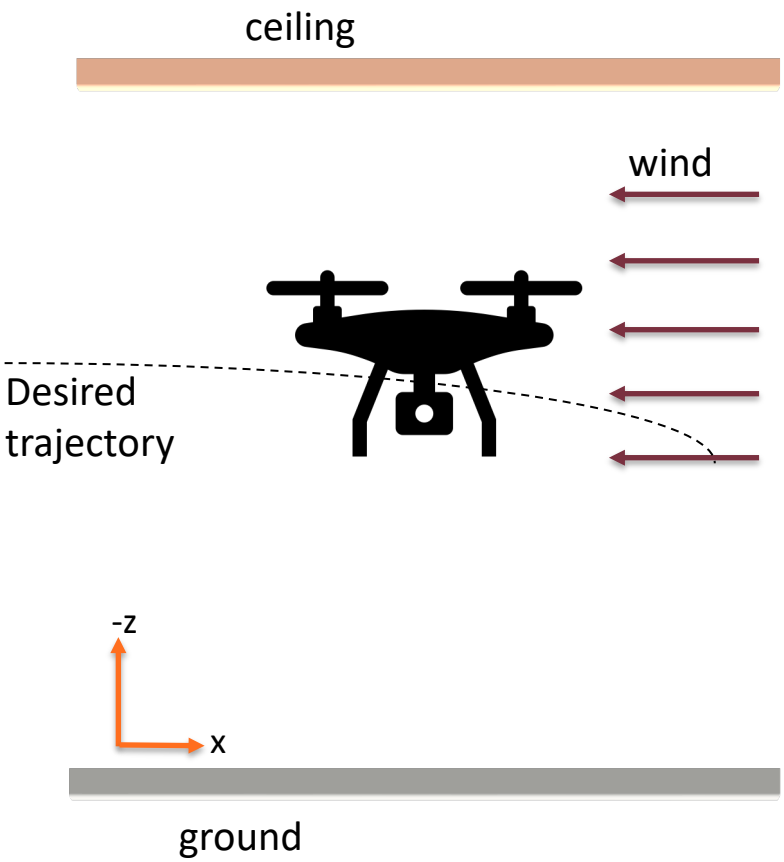
Discretized dynamics:

$$\boldsymbol{x}(t+1) = \boldsymbol{A}_d\boldsymbol{x}(t) + \boldsymbol{B}_d\boldsymbol{u}$$

Desired trajectory: $\boldsymbol{x}_{des}(t)$

Trajectory tracking error:

$$(\boldsymbol{x}(t) - \boldsymbol{x}_{des}(t))^T(\boldsymbol{x}(t) - \boldsymbol{x}_{des}(t))$$



Caltech

Problem Formulation

Consider the linear, discrete-time system at time k,

$$x(k+1) = Ax(k) + Bu(k)$$

where, $x \in \mathbb{R}^{n_x}, u \in \mathbb{R}^{n_u}, A \in \mathbb{R}^{n_x \times n_x}, B \in \mathbb{R}^{n_x \times n_u}$.

Suppose you want to find a sequence of control inputs

$$u_{0:N-1} = \{u_0, u_1, \ldots, u_{N-1}\}$$

and a corresponding sequence of states

$$x_{0:N} = \{x_0, x_1, \ldots, x_N\}$$

such that the following cost is minimized

$$J(x(0), U_0) = x_N^T P x_N + \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i \right)$$

Caltech

Problem Formulation

The cost is composed of two parts:

- Stage cost
- Cost-to-go

$$J(x(0), U_0) = x_N^T P x_N + \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i \right)$$

The stage cost is the cost of deviating from the desired state and control over the planning horizon, i.e., from time 0 → N-1.

The cost-to-go (or terminal cost) is the cost of the problem from time N → ∞. We will later look at ways to find the "best" terminal cost.

Unconstrained Optimal Control Problem (OCP)

Horizon

$$\min_{U_0} \quad x_N^T P x_N + \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i \right)$$

Cost

$$\forall k \in \{0, \dots N-1\}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k$$

Dynamics

$$x_0 = x(0)$$

Initial Condition

where, $P \succeq 0$ is the terminal weight,

$Q \succeq 0$ is the state weight, and

$R \succ 0$ is the input/control weight.

Caltech

Unconstrained Optimal Control Problem (OCP)

Horizon

$$\min_{U_0} \quad x_N^T P x_N + \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i \right)$$

Cost

$$\forall k \in \{0, \dots N-1\}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k$$

Dynamics

$$x_0 = x(0)$$

Initial Condition

where, $P \succeq 0$ is the terminal weight,

$Q \succeq 0$ is the state weight, and

$R \succ 0$ is the input/control weight.

We will look at **two approaches** to solve this problem:

1. Batch Approach
2. Recursive Approach (based on Dynamic Programming).

Caltech

1. Batch Approach: Setup

We write the dynamics constraints (equality constraints) in terms of the initial condition and the control input as,

$$
\begin{aligned}
x_1 &= Ax_0 + Bu_0, \\
x_2 &= Ax_1 + Bu_1 \\
&= A(Ax_0 + Bu_0) + Bu_1 \\
&= A^2x_0 + ABu_0 + Bu_1 \\
&\vdots \\
x_N &= A^N x_0 + \begin{bmatrix} A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix} \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}
\end{aligned}
$$

1. Batch Approach: Setup

We write the dynamics constraints (equality constraints) in terms of the initial condition and the control input as,

$$\underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}}_{X_0} = \underbrace{\begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\mathcal{S}_x} x(0) + \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}}_{\mathcal{S}_u} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{U_0}$$

Hence, all the dynamics constraints can be written in batch form as,

$$X_0 = \mathcal{S}_x x(0) + \mathcal{S}_u U_0$$

Also let, $\bar{Q} = \text{blkdiag}(\underbrace{Q, Q, \dots, Q}_{\text{N times}}, P)$ and $\bar{R} = \text{blkdiag}(\underbrace{R, R, \dots, R}_{\text{N times}})$

Caltech

1. Batch Approach: How to solve

Let's revisit the original optimization problem that we want to solve,

$$\begin{aligned} \min_{U_0} \quad & x_N^T P x_N + \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i \right) \\ \text{s.t.} \quad & x_{k+1} = A x_k + B u_k \\ & x_0 = x(0) \end{aligned}$$

We can rewrite the cost as, $J(x(0), U_0) = X_0^T \bar{Q} X_0 + U_0^T \bar{R} U_0$
and the equality constraints as $X_0 = \mathcal{S}_x x(0) + \mathcal{S}_u U_0$.

Hence, we can substitute the the constraints into the cost to get,

$$\min_{U_0} \quad (\mathcal{S}_x x(0) + \mathcal{S}_u U_0)^T \bar{Q} (\mathcal{S}_x x(0) + \mathcal{S}_u U_0) + U_0^T \bar{R} U_0$$

Caltech

1. Batch Approach: How to solve

The unconstrained optimization given by,

$$\min_{U_0} \quad (\mathcal{S}_x x(0) + \mathcal{S}_u U_0)^T \bar{Q}(\mathcal{S}_x x(0) + \mathcal{S}_u U_0) + U_0^T \bar{R} U_0$$

can be solved by taking the gradient of the cost,

$$\begin{aligned} 0 &= \nabla_{U_0} \Big((\mathcal{S}_x x(0) + \mathcal{S}_u U_0)^T \bar{Q}(\mathcal{S}_x x(0) + \mathcal{S}_u U_0) + U_0^T \bar{R} U_0 \Big) \\ &= \nabla_{U_0} \Big(x(0)^T \mathcal{S}_x^T \bar{Q} \mathcal{S}_x x(0) + 2x(0)^T \mathcal{S}_x^T \bar{Q} \mathcal{S}_u U_0 + U_0^T \mathcal{S}_u^T \bar{Q} \mathcal{S}_u U_0 + U_0^T \bar{R} U_0 \Big) \\ &= 2\big(x(0)^T \mathcal{S}_x^T \bar{Q} \mathcal{S}_u\big)^T + 2(\mathcal{S}_u^T \bar{Q} \mathcal{S}_u + \bar{R}) U_0. \end{aligned}$$

Hence, the solution of the optimization is given in closed-form as,

$$\boxed{U_0^*\big(x(0)\big) = -(\mathcal{S}_u^T \bar{Q} \mathcal{S}_u + \bar{R})^{-1} \mathcal{S}_u^T \bar{Q} \mathcal{S}_x x(0)}$$

Caltech

1. Batch Approach: Summary

We wrote the states in terms of the initial condition and the control inputs.

We were able to find the solution to the optimization because the cost was a convex, quadratic function of the optimization variables U_0.

The solution of the unconstrained optimal control problem is given in closed-form as,

$$U_0^*\big(x(0)\big) = -(\mathcal{S}_u^T \bar{Q} \mathcal{S}_u + \bar{R})^{-1} \mathcal{S}_u^T \bar{Q} \mathcal{S}_x x(0)$$

We know that $\mathcal{S}_u^T \bar{Q} \mathcal{S}_u + \bar{R}$ is invertible because $\mathcal{S}_u^T \bar{Q} \mathcal{S}_u \succeq 0$ & $\bar{R} \succ 0$.

If there are state or input constraints, solving this problem by matrix inversion is not guaranteed to result in a feasible input sequence.

Caltech

2. Recursive Approach: 1-step setup

Instead of the batch approach, we can also take a Dynamic Programming approach to solving the unconstrained, finite-horizon OCP.

Let's first look at the one-step optimal control problem at time k = N-1:

$$
\begin{aligned}
J^*_{N-1}(x_{N-1}) = \min_{u_{N-1}} \quad & x_N^T P_N x_N + x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} \\
\text{s.t.} \quad & x_N = A x_{N-1} + B u_{N-1} \\
& P_N = P
\end{aligned}
$$

where, we use P_j to denote the optimal cost-to-go and we set $P_N = P$.

2. Recursive Approach: 1-step solution

We will now solve this one-step problem, the same way we did earlier, by substituting the equality constraints,

$$J^*_{N-1}(x_{N-1}) = \min_{u_{N-1}} \; (Ax_{N-1} + Bu_{N-1})^T P_N (Ax_{N-1} + Bu_{N-1}) + x^T_{N-1} Q x_{N-1} + u^T_{N-1} R u_{N-1}$$

and then setting the gradient of the unconstrained minimization to 0,

$$\nabla_{u_{N-1}} \Big((Ax_{N-1} + Bu_{N-1})^T P_N (Ax_{N-1} + Bu_{N-1}) + x^T_{N-1} Q x_{N-1} + u^T_{N-1} R u_{N-1} \Big) = 0$$

to obtain the optimal control input at time N-1,

$$\begin{aligned} u^*_{N-1} &= -(B^T P_N B + R)^{-1} B^T P_N A x_{N-1} \\ &\triangleq F_{N-1} x_{N-1}. \end{aligned}$$

This optimal control input can by substituted into the cost to obtain:

$$\begin{aligned} J^*_{N-1}(x_{N-1}) &= x^T_{N-1} \big(A^T P_N A + Q - A^T P_N B (B^T P_N B + R)^{-1} B^T P_N A \big) x_{N-1} \\ &\triangleq x^T_{N-1} P_{N-1} x_{N-1} \end{aligned}$$

Caltech

2. Recursive Approach: 2-step setup

Now, we look at the two-step OCP at time k = N-2,

$$J^*_{N-2}(x_{N-2}) = \min_{u_{N-1},u_{N-2}} \; x_N^T P_N x_N + \sum_{i=N-2}^{N-1} x_i^T Q x_i + u_i^T R u_i$$

$$\text{s.t.} \quad x_N = Ax_{N-1} + Bu_{N-1}$$

$$x_{N-1} = Ax_{N-2} + Bu_{N-2}.$$

By **Bellman's Principle of Optimality**, the equivalent OCP is given by,

$$J^*_{N-2}(x_{N-2}) = \min_{u_{N-2}} \; J^*_{N-1}(x_{N-1}) + x_{N-2}^T Q x_{N-2} + u_{N-2}^T R u_{N-2}$$

$$\text{s.t.} \quad x_{N-1} = Ax_{N-2} + Bu_{N-2}$$

where we previously calculated the terminal cost,

$$J^*_{N-1}(x_{N-1}) = x_{N-1}^T \left(A^T P_N A + Q - A^T P_N B (B^T P_N B + R)^{-1} B^T P_N A\right) x_{N-1}$$

$$\triangleq x_{N-1}^T P_{N-1} x_{N-1}$$

Caltech

2. Recursive Approach: Solution

The two-step OCP below is solved the same way as the one-step OCP,

$$J^*_{N-2}(x_{N-2}) = \min_{u_{N-2}} \; x^T_{N-1} P_{N-1} x_{N-1} + x^T_{N-2} Q x_{N-2} + u^T_{N-2} R u_{N-2}$$
$$\text{s.t.} \quad x_{N-1} = A x_{N-2} + B u_{N-2}$$

to get the optimal control input,

$$u^*_{N-2} = -(B^T P_{N-1} B + R)^{-1} B^T P_{N-1} A x_{N-2}$$
$$\triangleq F_{N-2} x_{N-2}.$$

Similarly, the recursive solution evaluated for $k = \{N-1, \ldots, 0\}$ is,

$$u^*(k) = -(B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A x(k)$$
$$\triangleq F_k x(k).$$

and the recursive update of the terminal weight (also called the Discrete Time Riccati Equation) is,

$$P_k = A^T P_{k+1} A + Q - A^T P_{k+1} B (B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A$$
$$P_N = P$$

Caltech

2. Recursive Approach: Summary

We can solve the OCP in a recursive manner by starting at the last time step, k = N-1, working all the way up to k = 0.

Using the Principle of Optimality, the recursive solution is

$$u^*(k) = -(B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A x(k) \quad \forall k \in \{N-1, \ldots, 0\}$$
$$\triangleq F_k x(k).$$

and the recursive update of the terminal weight (also called the Discrete Time Riccati Equation) is,

$$P_k = A^T P_{k+1} A + Q - A^T P_{k+1} B (B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A$$
$$P_N = P$$

The optimal cost-to-go k → N is,

$$J^*(x(k)) = x(k)^T P_k x(k)$$

Caltech

Comparison of Batch and Recursive Approaches

To summarize, the solution of the **Batch Approach** is,

$$U_0^*\big(x(0)\big) = -(\mathcal{S}_u^T \bar{Q} \mathcal{S}_u + \bar{R})^{-1} \mathcal{S}_u^T \bar{Q} \mathcal{S}_x x(0)$$

and the solution of the **Recursive Approach** is,

$$\begin{aligned} u^*(k) &= -(B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A x(k) \\ &\triangleq F_k x(k), \\ P_k &= A^T P_{k+1} A + Q - A^T P_{k+1} B (B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A \\ P_N &= P \end{aligned}$$

Caltech

Comparison of Batch and Recursive Approaches

To summarize, the solution of the **Batch Approach** is,

$$U_0^*\big(x(0)\big) = -(\mathcal{S}_u^T \bar{Q} \mathcal{S}_u + \bar{R})^{-1} \mathcal{S}_u^T \bar{Q} \mathcal{S}_x x(0)$$

and the solution of the **Recursive Approach** is,

$$\begin{aligned} u^*(k) &= -(B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A x(k) \\ &\triangleq F_k x(k), \\ P_k &= A^T P_{k+1} A + Q - A^T P_{k+1} B (B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A \\ P_N &= P \end{aligned}$$

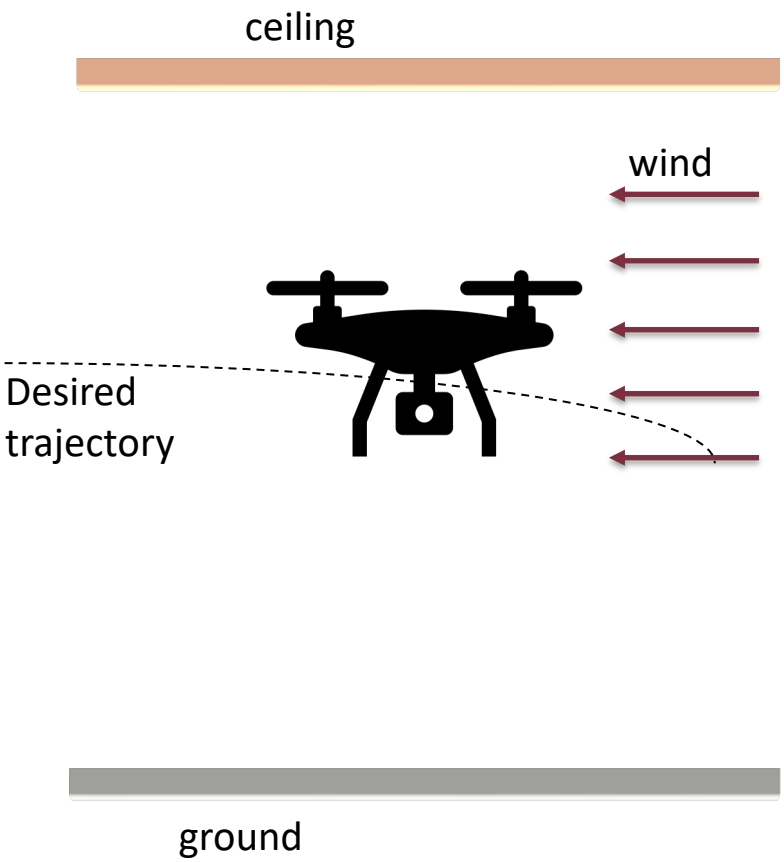
The difference between the two approaches is that the batch approach gives a *numerical sequence* of control inputs (open-loop sequence) whereas the recursive approach gives us a solution that is a *function of the state* at time k (closed-loop policy).

Caltech

Comparison of Batch and Recursive Approaches

The difference between the two approaches is that the batch approach gives a *numerical sequence* of control inputs (open-loop sequence) whereas the recursive approach gives us a solution that is a *function of the state* at time k (feedback policy).

If the state evolves exactly as modeled, there is no difference between the two approaches.



Caltech

Comparison of Batch and Recursive Approaches

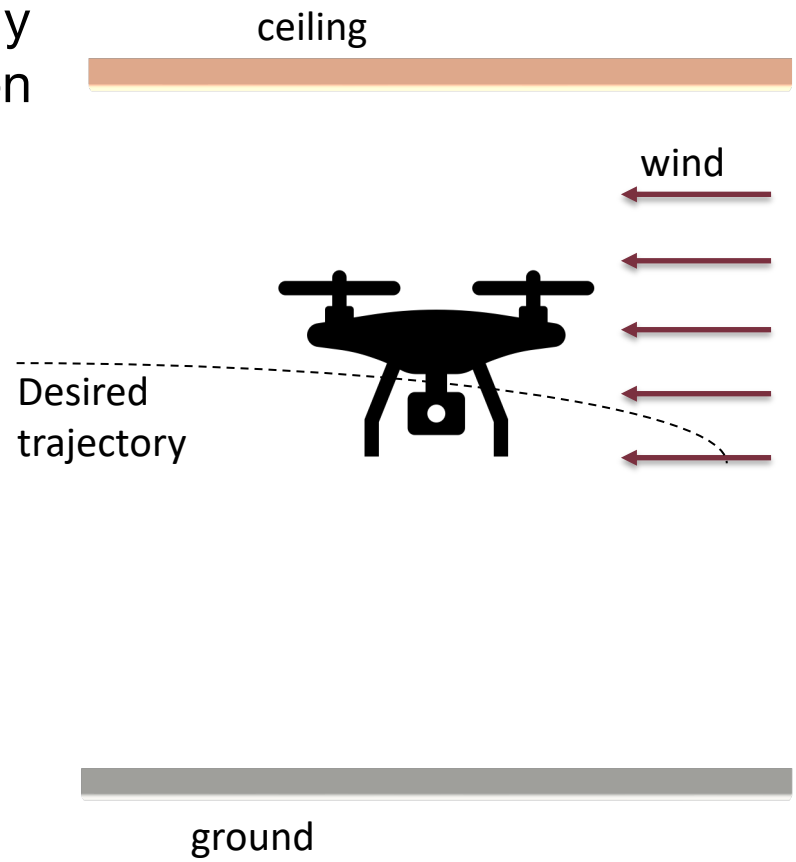
In the presence of any disturbances, the recursive approach can adapt more easily because the policy is computed based on the current state of the system and not just as a function of the initial state.

As the horizon length increases, the size of the matrix to be inverted for the batch approach increases and the recursive approach becomes more desirable.

Constraints can be adapted more easily using the batch approach.

Next Lecture:

How to incorporate constraints?



Caltech