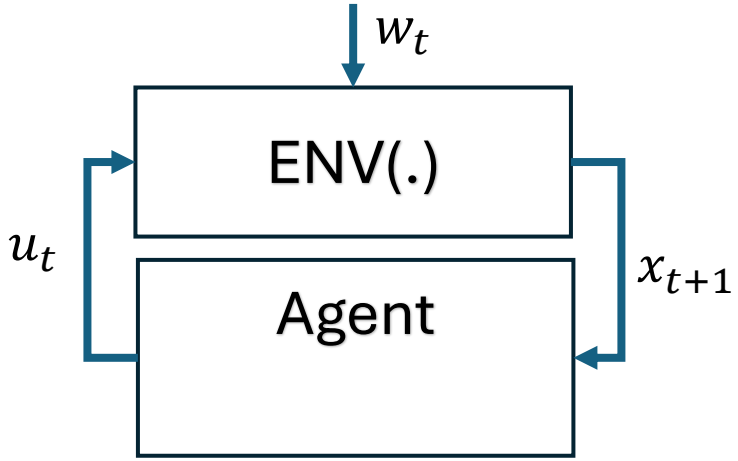
Semidefinite Programming for Domain Randomization in LQR Control

Abbas Pasdar
and
Farnaz Adib Yaghmaie

20 May 2026

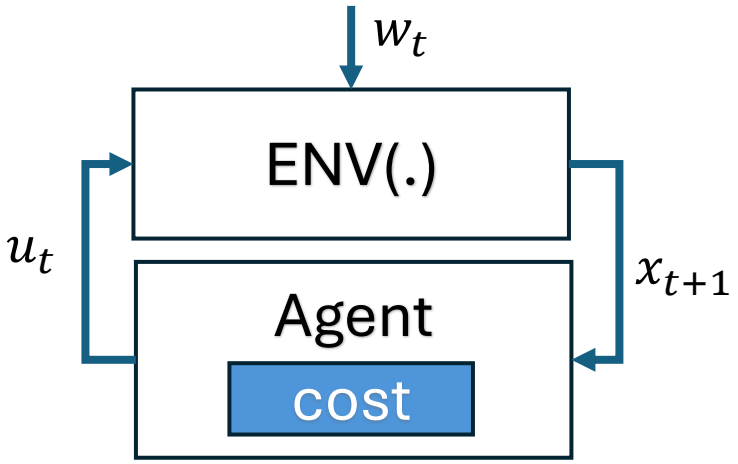
Introduction: Uncertainty in Control

- $x_{t+1} = ENV(u_t, w_t)$



Introduction: Uncertainty in Control

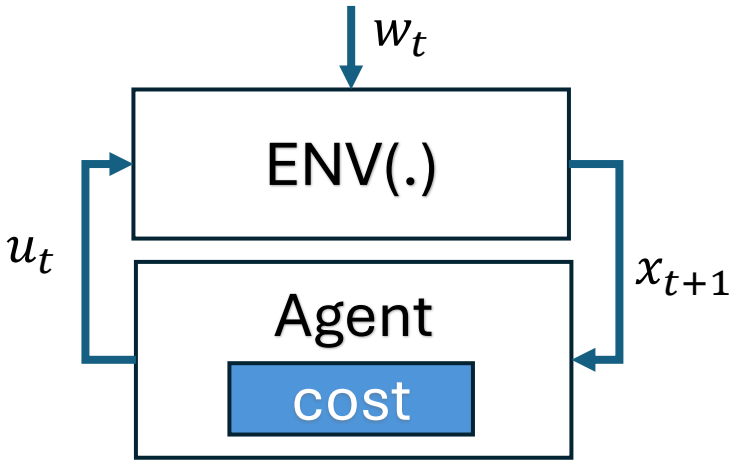
- $x_{t+1} = ENV(u_t, w_t)$
- The agent defines the performance function.



Introduction: Uncertainty in Control

- $x_{t+1} = ENV(u_t, w_t)$
- The agent defines the performance function.
- **Goal:** find optimal policy: $u_t = \pi(x_t)$.

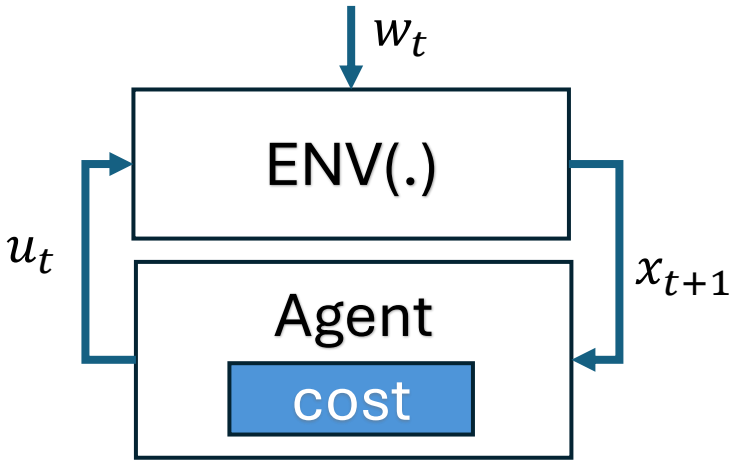
$$\pi^* = \arg\min_{\pi} \; J(\pi) := \sum_{t=0}^{\infty} c_t(x_t, u_t)$$



Introduction: Uncertainty in Control

- $x_{t+1} = ENV(u_t, w_t)$
- The agent defines the performance function.
- **Goal:** find optimal policy: $u_t = \pi(x_t)$.

$$\pi^* = \arg\min_{\pi} J(\pi) := \sum_{t=0}^{\infty} c_t(x_t, u_t)$$

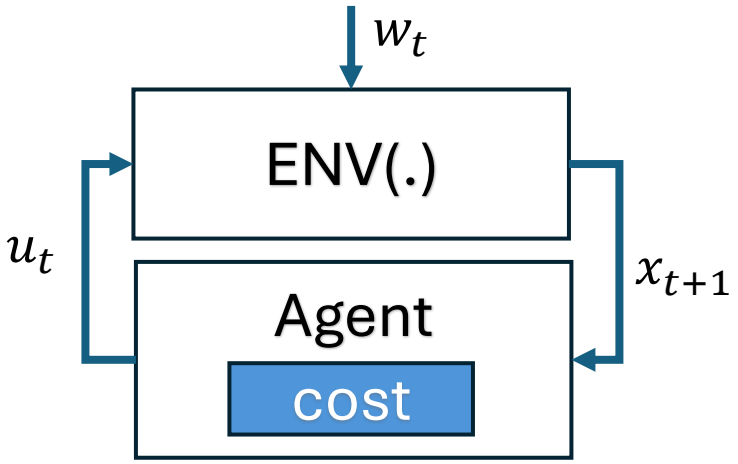


- **A convenient approach:** Simulation $x_{t+1} = f_\theta(x_t, u_t) + w_t$

Introduction: Uncertainty in Control

- $x_{t+1} = ENV(u_t, w_t)$
- The agent defines the performance function.
- **Goal:** find optimal policy: $u_t = \pi(x_t)$.

$$\pi^* = \arg\min_{\pi} J(\pi) := \sum_{t=0}^{\infty} c_t(x_t, u_t)$$

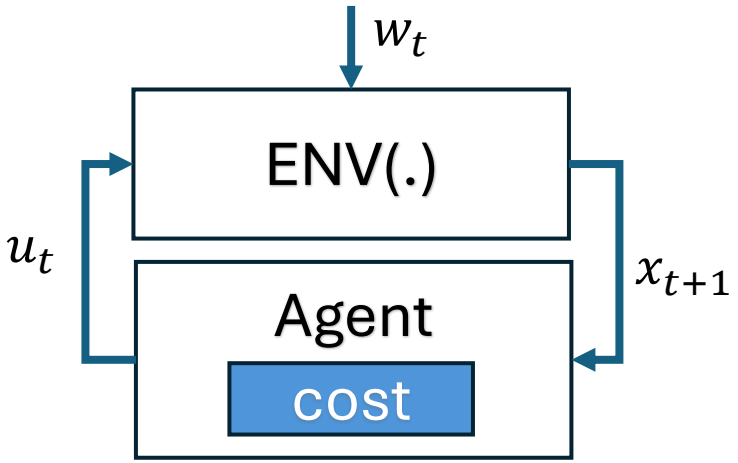


- **A convenient approach:** Simulation $x_{t+1} = f_\theta(x_t, u_t) + w_t$
- **Sim-to-reality problem:** θ cab be wrong ➔ Performance degradation.

Introduction: Uncertainty in Control

- $x_{t+1} = ENV(u_t, w_t)$
- The agent defines the performance function.
- **Goal:** find optimal policy: $u_t = \pi(x_t)$.

$$\pi^* = \arg\min_{\pi} \; J(\pi) := \sum_{t=0}^{\infty} c_t(x_t, u_t)$$

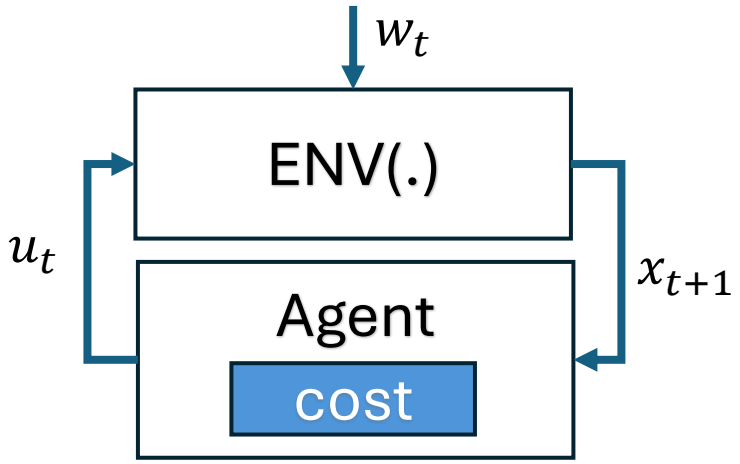


- **A convenient approach:** Simulation $x_{t+1} = f_\theta(x_t, u_t) + w_t$
- **Sim-to-reality problem:** θ cab be wrong ➔ Performance degradation.
- **Domain Randomization:** Assume $\theta \sim \mathcal{D}$, optimize $\pi^*_{DR} = \arg\min_{\pi} \; \mathbb{E}_\theta[J(\pi, \theta)]$

Introduction: Uncertainty in Control

- $x_{t+1} = ENV(u_t, w_t)$
- The agent defines the performance function.
- **Goal:** find optimal policy: $u_t = \pi(x_t)$.

$$\pi^* = \arg\min_{\pi} \; J(\pi) := \sum_{t=0}^{\infty} c_t(x_t, u_t)$$



- **A convenient approach:** Simulation $x_{t+1} = f_\theta(x_t, u_t) + w_t$
- **Sim-to-reality problem:** θ cab be wrong ➔ Performance degradation.
- **Domain Randomization:** Assume $\theta \sim \mathcal{D}$, optimize $\pi^*_{DR} = \arg\min_{\pi} \; \mathbb{E}_\theta[J(\pi, \theta)]$
- **Approximate by Monte Carlo**

Problem Formulation (DR-LQR)

Linear system:

$$\begin{cases} x_{t+1} = A_\theta x_t + B_\theta u_t + w_t \\ z_t = \begin{bmatrix} Q^{1/2} & 0 \\ 0 & R^{1/2} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} \\ c_t = z_t^\intercal z_t \end{cases}$$

Consider:

$$\begin{cases} \mathbb{E}[w_t] = 0 \\ \mathbb{E}[w_t^\intercal w_t] = I \end{cases}$$

- Optimal policy for conventional LQR: $u_t = \pi(x_t) = Kx_t$

Problem Formulation (DR-LQR)

Linear system:
$$\begin{cases} x_{t+1} = A_\theta x_t + B_\theta u_t + w_t \\ z_t = \begin{bmatrix} Q^{1/2} & 0 \\ 0 & R^{1/2} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} \\ c_t = z_t^\intercal z_t \end{cases}$$

Consider:
$$\begin{cases} \mathbb{E}[w_t] = 0 \\ \mathbb{E}[w_t^\intercal w_t] = I \end{cases}$$

- Optimal policy for conventional LQR: $u_t = \pi(x_t) = Kx_t$
- **DR-LQR:** Given M sample θ_j i.i.d, find

$$K^* = \arg\min_{K \in \mathcal{K}_{js}} J_{SA}(K) = \frac{1}{M} \sum_{j=1}^{M} \mathrm{Tr}\left[(Q + K^\intercal R K)\Sigma_j\right]$$
$$s.t. \;\; \Sigma_j = (A_j + B_j K)\Sigma_j (A_j + B_j K)^\intercal + I$$

Problem Formulation (DR-LQR)

Linear system:
$$\begin{cases} x_{t+1} = A_\theta x_t + B_\theta u_t + w_t \\ z_t = \begin{bmatrix} Q^{1/2} & 0 \\ 0 & R^{1/2} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} \\ c_t = z_t^\intercal z_t \end{cases}$$

Consider:
$$\begin{cases} \mathbb{E}[w_t] = 0 \\ \mathbb{E}[w_t^\intercal w_t] = I \end{cases}$$

- Optimal policy for conventional LQR: $u_t = \pi(x_t) = Kx_t$
- **DR-LQR:** Given M sample θ_j i.i.d, find

$$K^* = \arg\min_{K \in \mathcal{K}_{js}} J_{SA}(K) = \frac{1}{M} \sum_{j=1}^{M} \mathrm{Tr}\left[(Q + K^\intercal R K)\Sigma_j\right]$$
$$s.t. \;\; \Sigma_j = (A_j + B_j K)\Sigma_j (A_j + B_j K)^\intercal + I$$

- The set of Joint stabilizing controller: $\mathcal{K}_{js} := \{K: \; \rho(A_j + B_j K) < 1, \forall j\}$
- **Assumption:** $\mathcal{K}_{js}$ is non-empty.

Key Idea

- **One popular approach:** Policy Gradient (PG), but:
  - Do not inherently enforce stability for every model
  - Weak constraint handling

Key Idea

- **One popular approach:** Policy Gradient (PG), but:
  - Do not inherently enforce stability for every model
  - Weak constraint handling

- **Our proposal:** Use Semi-Definite Programming (SDP) Framework
  - Efficiently incorporate constraints such as stability.
  - Sample efficient in a data-driven method

Key Idea

- **One popular approach:** Policy Gradient (PG), but:
  - Do not inherently enforce stability for every model
  - Weak constraint handling

- **Our proposal:** Use Semi-Definite Programming (SDP) Framework
  - Efficiently incorporate constraints such as stability.
  - Sample efficient in a data-driven method

- **Challenge:** No direct SDP formulation exist for $M > 1$

Key Idea

- **One popular approach:** Policy Gradient (PG), but:
  - ➢ Do not inherently enforce stability for every model
  - ➢ Weak constraint handling

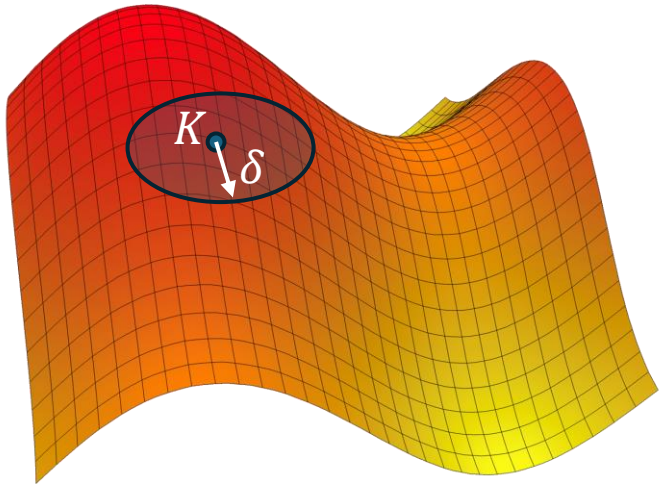
- **Our proposal:** Use Semi-Definite Programming (SDP) Framework
  - ➢ Efficiently incorporate constraints such as stability.
  - ➢ Sample efficient in a data-driven method

- **Challenge:** No direct SDP formulation exist for $M > 1$

- **Idea:** Iteratively update K solving SDP problems ➔ **SDP descent**

Proposed Algorithm: SDPD

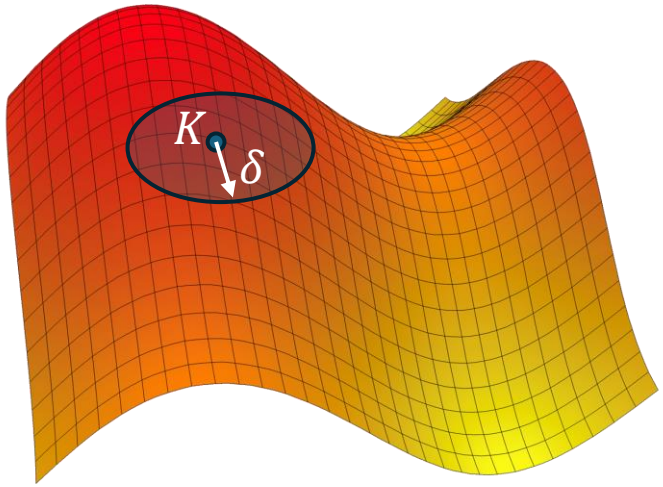
Given stabilizing gain K and Σ_j at iteration i:



Proposed Algorithm: SDPD

Given stabilizing gain K and Σ_j at iteration i:

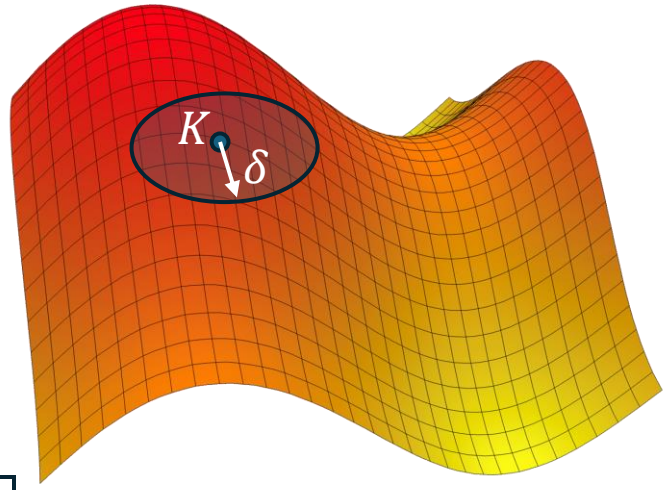
- Perturb by $K + \delta$ and $\Sigma_j + \sigma_j$
- Linearize objective and constraints by:
  - Expanding
  - Removing high order terms



Proposed Algorithm: SDPD

Given stabilizing gain K and Σ_j at iteration i:

- Perturb by $K + \delta$ and $\Sigma_j + \sigma_j$
- Linearize objective and constraints by:
  - Expanding
  - Removing high order terms
- Results in an SDP:



$$\delta^* = \arg\min_{\sigma_j, \delta} \sum_{j=1}^{M} \mathrm{Tr}\left[Q_K\, \sigma_j + R_{K,j}\, \delta\right]$$

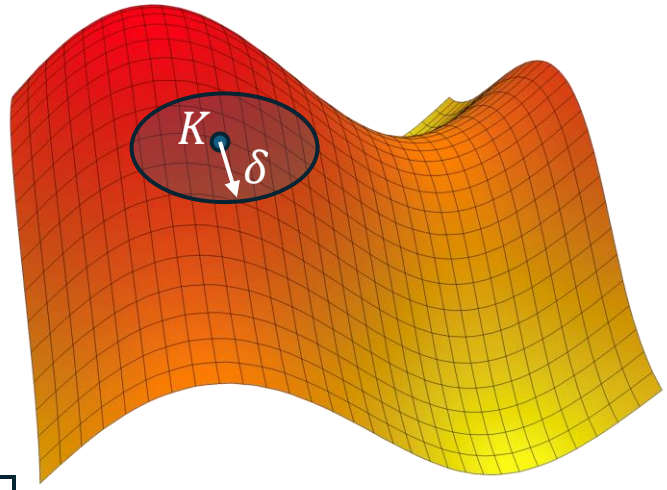
$$s.t. \quad \sigma_j = \bar{A}_j\, \sigma_j\, \bar{A}_j^{\intercal} + \bar{A}_j\, \Sigma_j \left(B_j \delta\right)^{\intercal} + B_j\, \delta\, \Sigma_j\, \bar{A}_j^{\intercal}$$

$$\Sigma_j + \sigma_j \succ 0, \qquad \|\delta\|_F < \eta, \qquad j = 1, \ldots, M$$

$\bar{A}_j = A_j + B_j K$, $Q_K = Q + K^{\intercal} R K$, and $R_{K,j} = 2\Sigma_j K^{\intercal} R$

Proposed Algorithm: SDPD

Given stabilizing gain K and Σ_j at iteration i:



- Perturb by $K + \delta$ and $\Sigma_j + \sigma_j$
- Linearize objective and constraints by:
  - Expanding
  - Removing high order terms
- Results in an SDP:

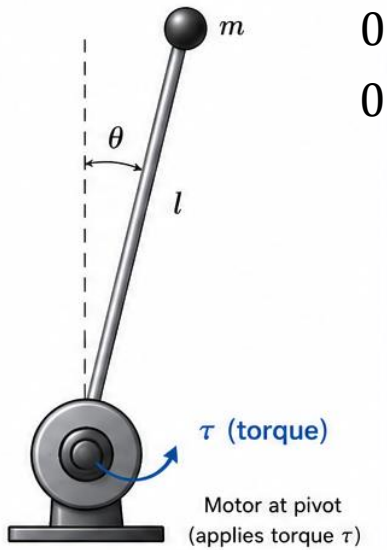
$$\delta^* = \arg\min_{\sigma_j, \delta} \sum_{j=1}^{M} \mathrm{Tr}\left[Q_K\, \sigma_j + R_{K,j}\, \delta\right]$$

$$s.t. \quad \sigma_j = \bar{A}_j\, \sigma_j\, \bar{A}_j^{\mathsf{T}} + \bar{A}_j\, \Sigma_j \left(B_j \delta\right)^{\mathsf{T}} + B_j\, \delta\, \Sigma_j\, \bar{A}_j^{\mathsf{T}}$$

$$\Sigma_j + \sigma_j \succ 0, \qquad \|\delta\|_F < \eta, \qquad j = 1, \ldots, M$$

- $\|\delta\|_F < \eta$: Stability and linearization guarantee
- δ^* is the steepest descent of J_{SA} in linearized constraint set.

Experiments

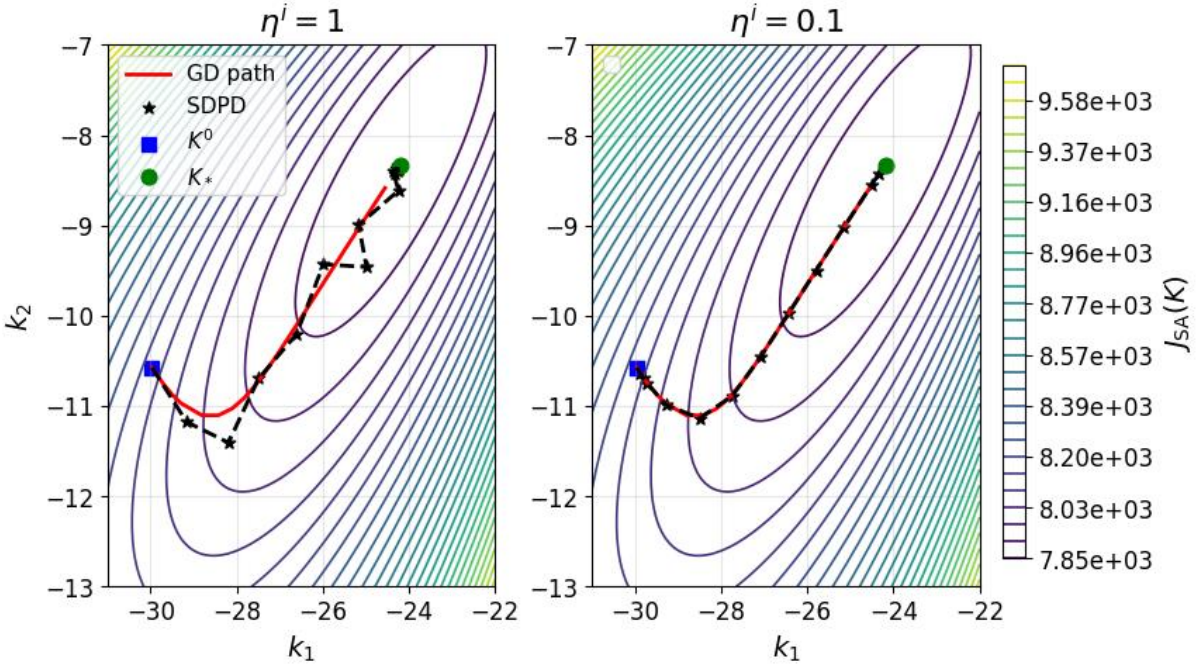


Linearized Discrete–Time Dynamics
(about the upright position)

$$A = \begin{bmatrix} 1 & \Delta t \\ \frac{g}{l}\Delta t & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{1}{ml^2}\Delta t \end{bmatrix}$$

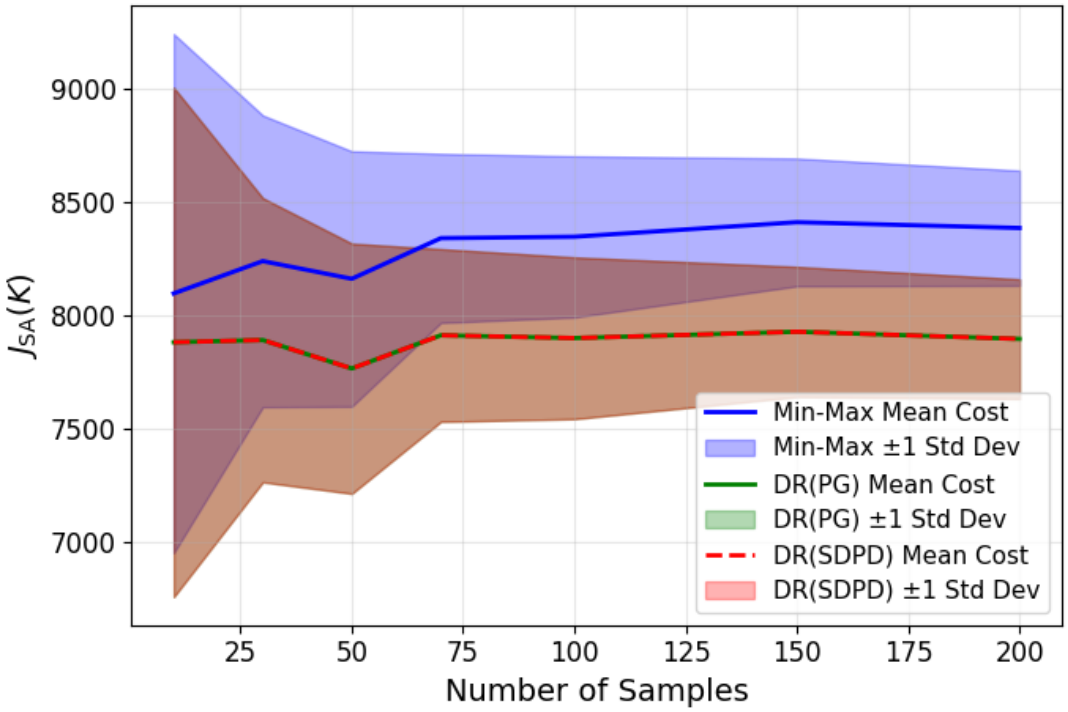
θ = angular displacement (rad)
$\dot{\theta}$ = angular velocity (rad/s)
τ = input torque (N·m)
$g = 9.81\ \mathrm{m/s^2}$
Δt = sampling time (s)

$$0.75 \leq m \leq 1.25$$
$$0.75 \leq l \leq 1.25$$



Experiments

- Domain Randomization vs Min-Max (100 try):



Summary

- **Domain Randomization:** model (environment) uncertainty
  - Less conservative than Min-Max
- **Use SDP for DR-LQR:**
  - Efficient for constraint optimization (e.g., stability constraints)
  - Sample efficient
- **SDPD:** Iteratively solve an SDP for the next update

Thank you for your attention.

Property

- **Proposition 1** (Stability):
  - For small enough update, $K + \delta$ remains stabilizing.
  - *A sufficient condition is:*

$$\|\delta\|_F < \eta = \min_j \frac{\sqrt{\|\bar{A}_j\|_F^2 + \|\Sigma_j\|_F^{-1}} - \|\bar{A}_j\|_F}{\|B_j\|_F}$$

- **Proposition 2** (linear approximation):
  - The covariance change satisfies $\|\sigma_j\|_F = \mathcal{O}(\|\delta\|_F)$,
  - The linear approximation is accurate for small steps.

- **In practice:**
  - choose η from the stability bounds.
  - Shrink the step if cost does not improve.

Joint Stabilizing Initial Controller (JSIC)

How to find K_0 stabilizing all sample systems?

1) Common state covariance constraint

$$\Sigma_j = \Sigma \text{ for all } j = 1, \dots, M$$

➔ SDP formulation exist

➔ Can become infeasible

2) K_0 is stabilizing for all sample systems if $\alpha^* < 1$ in:

$$\alpha^* = \arg \min_{\alpha_j, K_0, P_j} \alpha$$

$$s.t. \quad \left(A_j + B_j K_0\right) P_j \left(A_j + B_j K_0\right)^{\mathsf{T}} \prec \alpha_j P_j,$$

$$\alpha > \alpha_j, \quad P_j \succ 0, \quad j = 1, \dots, M$$

➔ **Use SDPD** starting with $K_0 = 0, \alpha_j = \left[(1+\varepsilon)\rho\left(A_j\right)\right]^2$