On the Sample Complexity of Stabilizing LTI Systems on a Single Trajectory

Yang Hu (Harvard), Adam Wierman (Caltech), Guannan Qu (CMU)

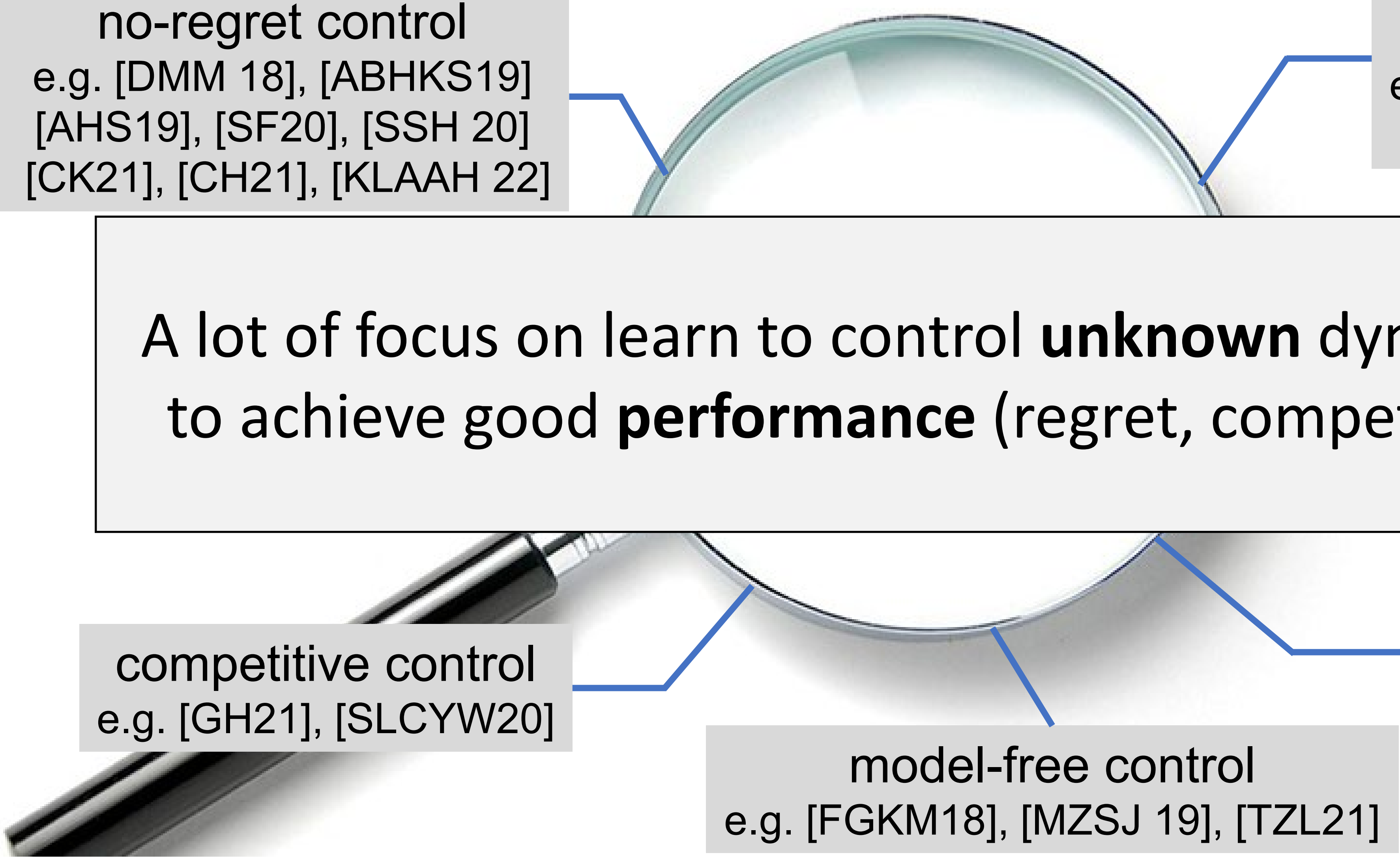
Presented by: Guannan Qu
Assistant Professor of ECE, Carnegie Mellon University

Sept 29, 2022

A Lot of Interest in Learning-based Control

Learning applied to control

A Lot of Interest in Learning-based Control



no-regret control
e.g. [DMM 18], [ABHKS19]
[AHS19], [SF20], [SSH 20]
[CK21], [CH21], [KLAAH 22]

sample complexity
e.g. [DMMRT18], [SMTJR18]
[SR 2019]

A lot of focus on learn to control **unknown** dynamical system to achieve good **performance** (regret, competitive ratio, …)

competitive control
e.g. [GH21], [SLCYW20]

model-free control
e.g. [FGKM18], [MZSJ 19], [TZL21]

dynamic regret with predictions/delays
e.g. [ZLL21], [LCL19]

Learn-to-*Stabilize* is Equally Important

Recent literature: Faradonbeh et al. 2019, Chen and Hazan 2020, Lale et al. 2020, Perdomo et al. 2021, Tsiamis et al. 2022.

Older adaptive control literature: Lai 1986, Chen and Zhang 1989, Lai and Ying 1991.

Problem Setup

Here we focus on the noiseless case.

Linear Time Invariant (LTI) System

unknown

$$x_{t+1} = \boxed{A}x_t + \boxed{B}u_t$$

One can choose $u_0, u_1, \ldots$ and observe $x_0, x_1, \ldots$

How to learn from data to stabilize the LTI system?

A Direct Attempt

Linear Time Invariant (LTI) System

$$x_{t+1} = Ax_t + Bu_t$$

One can choose $u_0, u_1, \ldots$ and observe $x_0, x_1, \ldots$

What about use these data to learn A, B…

… and then design a stabilizing controller?

- Variants of this idea has been adopted to solve the learn-to-stabilize problem, e.g. in [Chen and Hazan 2020], [Lale et al. 2020], etc.
    - This attempt is by no means simple, and various challenges including exploration, explosive trajectory, etc., needs to be addressed properly.

A Direct Attempt

$$x_{t+1} = Ax_t + Bu_t$$

Claim: state norm $\|x_t\|$ and regret can reach $2^{\Theta(n)}$ (n is dimension of state).

- Number of state samples $\{x_t\}$ to learn A, B scales linearly in n.

dimension of each x_t is n

dimension of A is n^2

It takes at least n state samples to obtain enough information needed to learn A.

A Direct Attempt

$$x_{t+1} = Ax_t + Bu_t$$

Claim: state norm $\|x_t\|$ and regret can reach $2^{\Theta(n)}$ (n is dimension of state).

- When collecting the n samples needed, the system can be unstable.

exponentially large state

$\|x_t\| \approx 2^{\Theta(n)}$

- The exponential regret is observed and justified in [Chen and Hazan 2020], etc.

Lower Bound

Theorem *(Chen and Hazan 2020, informal version)*

For any randomized algorithm, there exists a LTI instance such that the regret is lower bounded by

$$\text{regret} \geq 2^{\Omega(n)}$$

- Is learning the full matrix A, B really necessary for *stabilization*?
    - For example, if the system is *open-loop stable*, nothing needs to be learned!

- Are there instance-specific properties that allow us only learn partial information of A, B for stabilization?

Main Result *(informal version)*

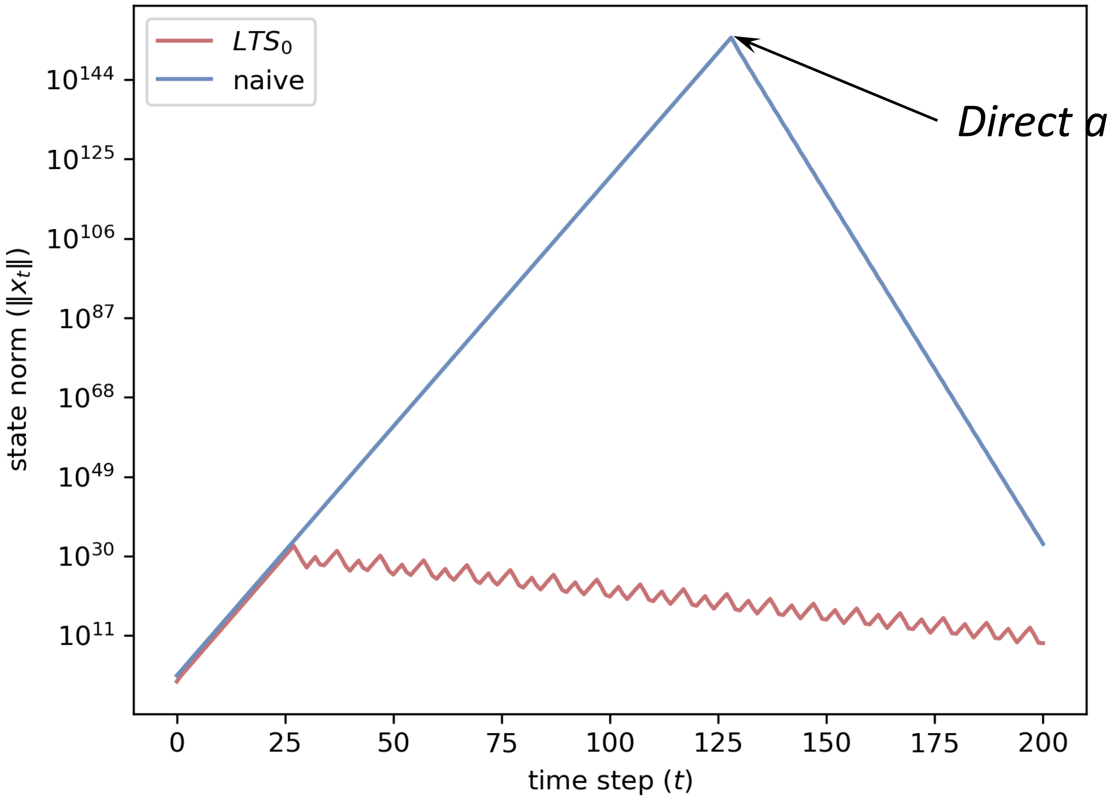
The proposed algorithm takes $O(k \log n)$ samples to stabilize the system, where k is the number of unstable eigenvalues.

Incurs a state norm of $2^{O(k \log n)}$

Avoids exponential blow-up $2^{\Theta(n)}$ *when* $k \ll n$

Can we exploit instance specific properties to learn to stabilize ***without incurring a state norm exponentially large in*** n*?*

$k = 3, n = 128$



Direct approach blows up to 10^{144}*!*

Can we exploit instance specific properties to learn to stabilize

without incurring a state norm exponentially large in n***?***

Summary of our approach

- **Ingredient 1: subspace decomposition**
  - We show only info about a k-dim subspace is needed for stabilization,
  - where k is the number of unstable eigenvalues.

- **Ingredient 2: subspace learning**
  - We design an algorithm using $O(k \log n)$ samples to learn the subspace.
  - This incurs a state norm $2^{O(k \log n)} \ll 2^{\Theta(n)}$ in the regime $k \ll n$.

Can we exploit instance specific properties to learn to stabilize
without incurring a state norm exponentially large in n?

Summary of our approach

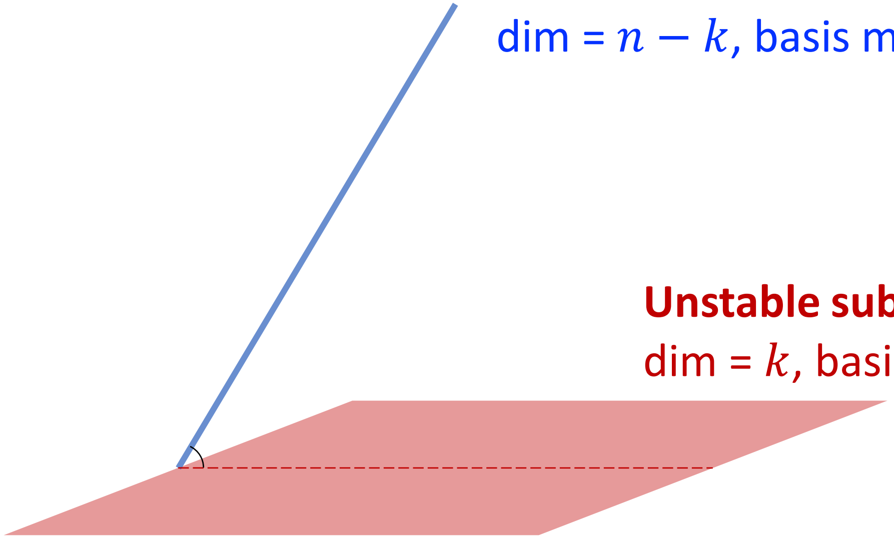
- **Ingredient 1: subspace decomposition**
    - We show only info about a k-dim subspace is needed for stabilization,
    - where k is the number of unstable eigenvalues.
- **Ingredient 2: subspace learning**
    - We design an algorithm using $O(k \log n)$ samples to learn the subspace.
    - This incurs a state norm $2^{O(k \log n)} \ll 2^{\Theta(n)}$ in the regime $k \ll n$.

Can we exploit instance specific properties to learn to stabilize ***without incurring a state norm exponentially large in n?***

Decomposition of Stable/Unstable Subspaces

$$x_{t+1} = Ax_t + Bu_t$$

Eigenvalues of A: $\overbrace{|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_k|}^{\textit{Unstable eigenvalues}} > 1 > \overbrace{|\lambda_{k+1}| \geq \cdots \geq |\lambda_n|}^{\textit{Stable eigenvalues}}$



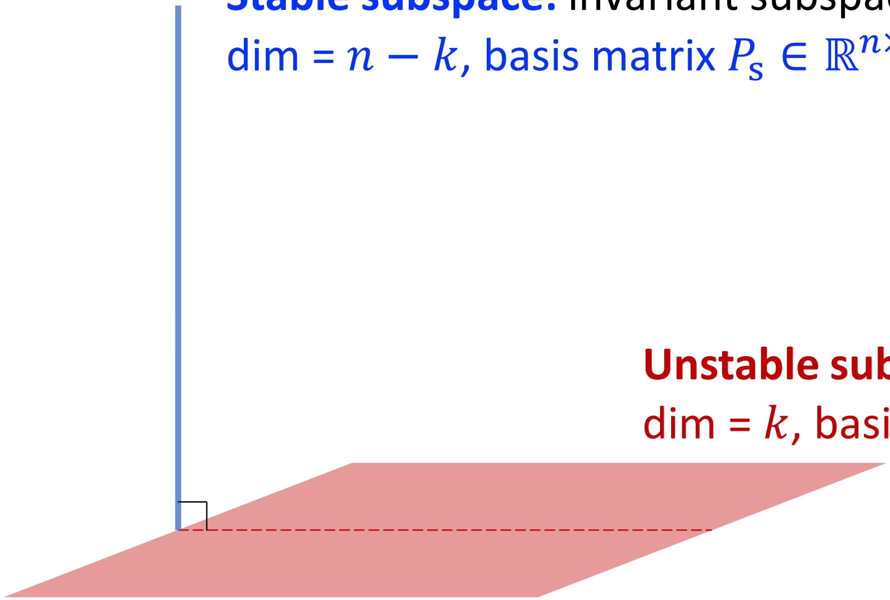
Stable subspace: invariant subspace of stable eigenvalues.
dim = $n - k$, basis matrix $P_\mathrm{s} \in \mathbb{R}^{n\times(n-k)}$.

Unstable subspace: invariant subspace of unstable eigenvalues.
dim = k, basis matrix $P_\mathrm{u} \in \mathbb{R}^{n\times k}$.

Decomposition of Stable/Unstable Subspaces

For illustration, consider the two subspaces are **orthogonal**.

(the non-orthogonal case will be dealt with later)



Stable subspace: invariant subspace of stable eigenvalues.

dim = $n-k$, basis matrix $P_{\mathrm{s}} \in \mathbb{R}^{n\times(n-k)}$.

Unstable subspace: invariant subspace of unstable eigenvalues.

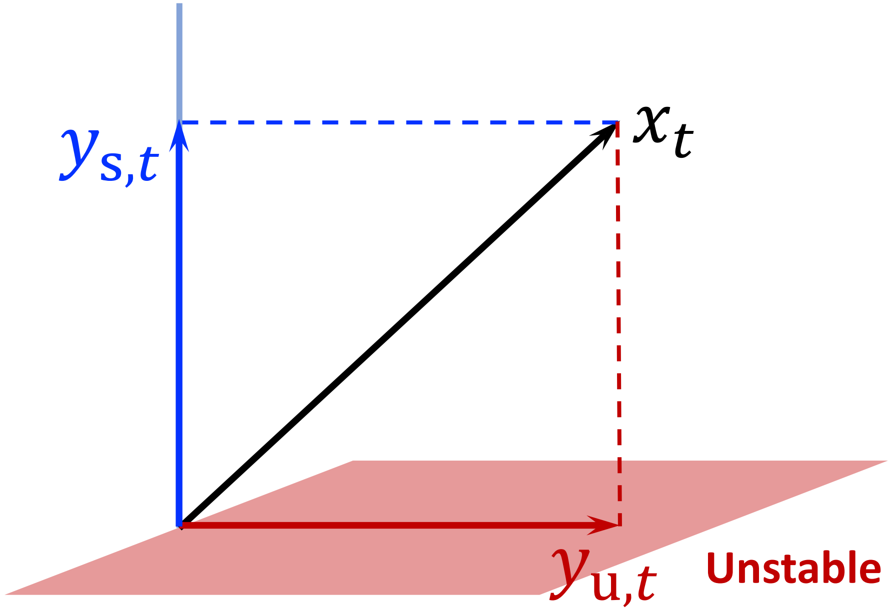
dim = k, basis matrix $P_{\mathrm{u}} \in \mathbb{R}^{n\times k}$.

Decomposition of Stable/Unstable Subspaces

Ingredient 1: information of k-dim unstable subspace is sufficient for stabilization.

$$\begin{bmatrix} y_{\mathrm{u},t+1} \\ y_{\mathrm{s},t+1} \end{bmatrix} = \begin{bmatrix} M_{\mathrm{u}} & 0 \\ 0 & M_{\mathrm{s}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix} + \begin{bmatrix} P_{\mathrm{u}}^{\top} B \\ P_{\mathrm{s}}^{\top} B \end{bmatrix} u_t$$

Stable subspace with basis $P_{\mathrm{s}} \in \mathbb{R}^{n\times(n-k)}$



$$\begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix} = \begin{bmatrix} P_{\mathrm{u}}^{\top} \\ P_{\mathrm{s}}^{\top} \end{bmatrix} x_t$$

Unstable subspace with basis $P_{\mathrm{u}} \in \mathbb{R}^{n\times k}$

Decomposition of Stable/Unstable Subspaces

Ingredient 1: information of k-dim unstable subspace is sufficient for stabilization.

$$\begin{bmatrix} y_{\mathrm{u},t+1} \\ y_{\mathrm{s},t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} M_{\mathrm{u}} & 0 \\ 0 & M_{\mathrm{s}} \end{bmatrix}} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix} + \begin{bmatrix} P_{\mathrm{u}}^{\top} B \\ P_{\mathrm{s}}^{\top} B \end{bmatrix} u_t$$

- This is block-diagonal because the subspaces are invariant w.r.t. A, and as such

has all unstable eigenvalues of A

$$[P_{\mathrm{u}}, P_{\mathrm{s}}]^{-1} A [P_{\mathrm{u}}, P_{\mathrm{s}}] = \begin{bmatrix} \overset{\downarrow}{M_{\mathrm{u}}} & 0 \\ 0 & \underset{\uparrow}{M_{\mathrm{s}}} \end{bmatrix}$$

has all stable eigenvalues of A

Decomposition of Stable/Unstable Subspaces

Ingredient 1: information of k-dim unstable subspace is sufficient for stabilization.

$$\begin{bmatrix} y_{\mathrm{u},t+1} \\ y_{\mathrm{s},t+1} \end{bmatrix} = \begin{bmatrix} M_{\mathrm{u}} & 0 \\ 0 & M_{\mathrm{s}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix} + \begin{bmatrix} P_{\mathrm{u}}^{\top} B \\ P_{\mathrm{s}}^{\top} B \end{bmatrix} u_t$$

setting $u_t = -K_{\mathrm{u}} y_{\mathrm{u},t}$

$$\begin{bmatrix} y_{\mathrm{u},t+1} \\ y_{\mathrm{s},t+1} \end{bmatrix} = \begin{bmatrix} M_{\mathrm{u}} - P_{\mathrm{u}}^{\top} B K_{\mathrm{u}} & 0 \\ -P_{\mathrm{s}}^{\top} B K_{\mathrm{u}} & M_{\mathrm{s}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix}$$

Decomposition of Stable/Unstable Subspaces

Ingredient 1: information of k-dim unstable subspace is sufficient for stabilization.

can be made stable
via a properly designed K_{u}

$$\begin{bmatrix} y_{\mathrm{u},t+1} \\ y_{\mathrm{s},t+1} \end{bmatrix} = \begin{bmatrix} \boxed{M_{\mathrm{u}} - P_{\mathrm{u}}^{\top} B K_{\mathrm{u}}} & 0 \\ -P_{\mathrm{s}}^{\top} B K_{\mathrm{u}} & \boxed{M_{\mathrm{s}}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix}$$

already a stable matrix

Decomposition of Stable/Unstable Subspaces

Ingredient 1: information of k-dim unstable subspace is sufficient for stabilization.

- P_{u}: basis of unstable subspace
- M_{u}: transition matrix of unstable component
- $P_{\mathrm{u}}^{\top} B$: projection of B onto unstable subspace.

can be made stable
via a properly designed K_{u}

$$\begin{bmatrix} y_{\mathrm{u},t+1} \\ y_{\mathrm{s},t+1} \end{bmatrix} = \begin{bmatrix} \boxed{M_{\mathrm{u}} - P_{\mathrm{u}}^{\top} B K_{\mathrm{u}}} & 0 \\ -P_{\mathrm{s}}^{\top} B K_{\mathrm{u}} & \boxed{M_{\mathrm{s}}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix}$$

already a stable matrix

Summary of our approach

- **Ingredient 1: subspace decomposition**
    - We show only info about a k-dim subspace is needed for stabilization,
    - where k is the number of unstable eigenvalues.

- **Ingredient 2: subspace learning**
    - We design an algorithm using $O(k \log n)$ samples to learn the subspace.
    - This incurs a state norm $2^{O(k \log n)} \ll 2^{\Theta(n)}$ in the regime $k \ll n$.

Can we exploit instance specific properties to learn to stabilize

without incurring a state norm exponentially large in n?

How to Learn P_u (basis of the unstable subspace)?

Idea: Open-loop system automatically drives states to the unstable subspace.

$$\begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix} = \begin{bmatrix} M_\mathrm{u} & 0 \\ 0 & M_\mathrm{s} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t-1} \\ y_{\mathrm{s},t-1} \end{bmatrix} = \begin{bmatrix} M_\mathrm{u}^t & 0 \\ 0 & M_\mathrm{s}^t \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},0} \\ y_{\mathrm{s},0} \end{bmatrix}$$

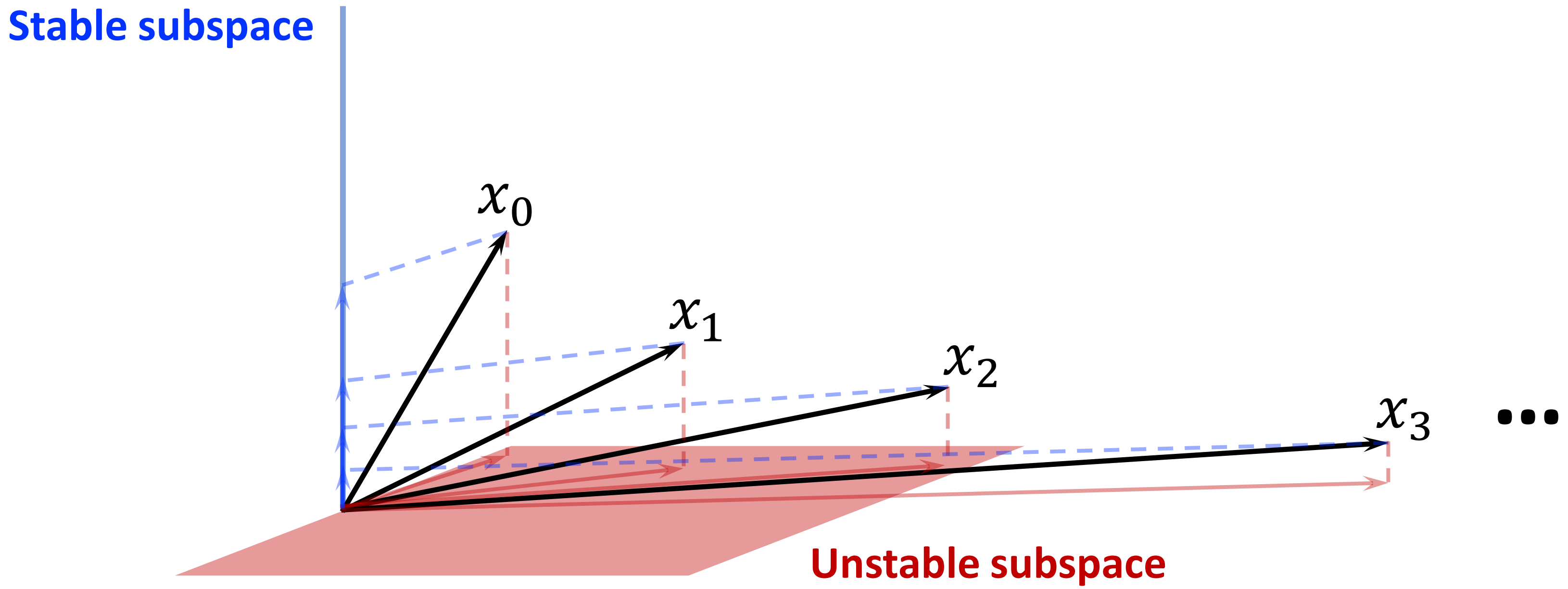
blows up

converges to 0

How to Learn P_{u} (basis of the unstable subspace)?

Idea: Open-loop system automatically drives states to the unstable subspace.

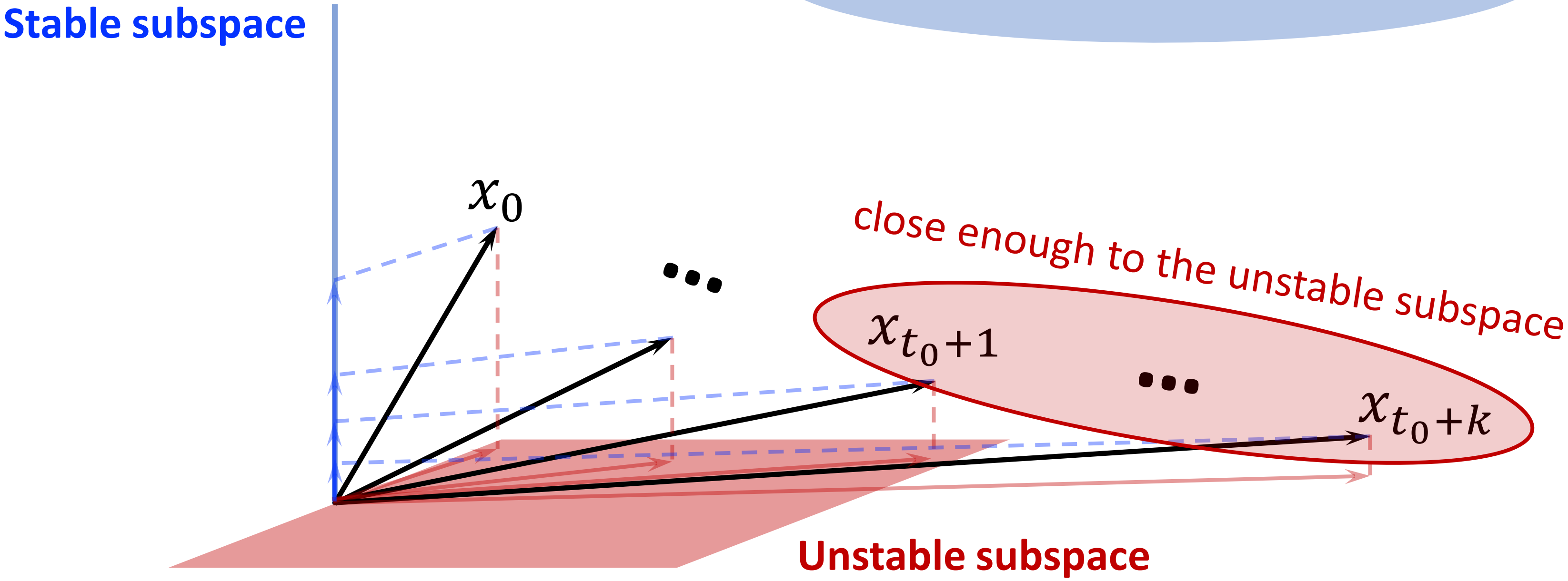
$$\begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix} = \begin{bmatrix} M_{\mathrm{u}} & 0 \\ 0 & M_{\mathrm{s}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t-1} \\ y_{\mathrm{s},t-1} \end{bmatrix} = \begin{bmatrix} M_{\mathrm{u}}^t & 0 \\ 0 & M_{\mathrm{s}}^t \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},0} \\ y_{\mathrm{s},0} \end{bmatrix}$$



How to Learn P_{u} (basis of the unstable subspace)?

Stage 1 of the Algorithm: learning P_{u}
Let the system run open-loop for a period of time $t_0 = O(k \log n)$.
Set $\widehat{P}_{\mathrm{u}}$ as an orthonormal basis of the subspace spanned by $[x_{t_0+1}, \dots, x_{t_0+k}]$.

This only takes k samples!



How to Learn M_{u}?

Recall: $\begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix} = \begin{bmatrix} M_{\mathrm{u}} & 0 \\ 0 & M_{\mathrm{s}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t-1} \\ y_{\mathrm{s},t-1} \end{bmatrix}$

$$\Rightarrow \quad y_{\mathrm{u},t} = M_{\mathrm{u}} y_{\mathrm{u},t-1}$$

$$\Rightarrow \quad P_{\mathrm{u}}^{\top} x_t = M_{\mathrm{u}} (P_{\mathrm{u}}^{\top} x_{t-1})$$

M_{u} can be obtained by least squares over the projected trajectory!

How to Learn M_{u}?

Stage 2 of the Algorithm: learning M_{u}

$$\widehat{M}_{\mathrm{u}} \leftarrow \arg \min_{\widehat{M}_{\mathrm{u}} \in \mathbb{R}^{k \times k}} \mathcal{L}(\widehat{M}_{\mathrm{u}}) := \sum_{t=t_0+1}^{t_0+k} \left\| \widehat{P}_{\mathrm{u}}^{\top} x_{t+1} - \widehat{M}_{\mathrm{u}} \widehat{P}_{\mathrm{u}}^{\top} x_t \right\|^2$$

This only takes $O(k)$ samples!

Full Algorithm (Orthogonal Case)

Stage 1: learning P_{u}

Let the system run open-loop for a period of time $t_0 = O(k \log n)$.

Set $\hat{P}_u$ as an orthonormal basis of the subspace spanned by $[x_{t_0+1}, \dots, x_{t_0+k}]$.

Stage 2: learning M_{u}

$$\hat{M}_{\mathrm{u}} \leftarrow \arg \min_{\hat{M}_{\mathrm{u}} \in \mathbb{R}^{k \times k}} \mathcal{L}(\hat{M}_{\mathrm{u}}) \coloneqq \sum_{t=t_0+1}^{t_0+k} \left\| \hat{P}_{\mathrm{u}}^{\top} x_{t+1} - \hat{M}_{\mathrm{u}} \hat{P}_{\mathrm{u}}^{\top} x_t \right\|^2$$

Stage 3: learning $B_{\mathrm{u}} \coloneqq P_{\mathrm{u}}^{\top} B$

Run the system for k more steps with input $u_{t_0+k+i} = \alpha \| x_{t_0+k+i} \| e_i$.

Let $\hat{B}_{\mathrm{u}} = [\hat{b}_1, \dots, \hat{b}_k]$, where $\hat{b}_i = \frac{1}{\alpha \| x_{t_0+k+i} \|} \left(\hat{P}_{\mathrm{u}}^{\top} x_{t_0+k+i+1} - \hat{M}_{\mathrm{u}} \hat{P}_{\mathrm{u}}^{\top} x_{t_0+k+i} \right)$.

Stage 4: design the stabilizing controller

Return state feedback controller $u = -\hat{K} x$, where $\hat{K} = -\hat{B}_{\mathrm{u}}^{-1} \hat{M}_{\mathrm{u}} \hat{P}_{\mathrm{u}}^{\top}$

Full Algorithm (Orthogonal Case)

Stage 1: learning P_{u}

Let the system run open-loop for a period of time $t_0 = O(k \log n)$.

Set $\widehat{P}_u$ as an orthonormal basis of the subspace spanned by $[x_{t_0+1}, \dots, x_{t_0+k}]$.

Stage 2: learning M_{u}

$$\widehat{M}_{\mathrm{u}} \leftarrow \arg \min_{\widehat{M}_{\mathrm{u}} \in \mathbb{R}^{k\times k}} \mathcal{L}(\widehat{M}_{\mathrm{u}}) := \sum_{t=t}^{t_0+k}$$

Stage 3: learning $B_{\mathrm{u}} := P_{\mathrm{u}}^{\top} B$

Run the system for k more steps

Let $\widehat{B}_{\mathrm{u}} = [\hat{b}_1, \dots, \hat{b}_k]$, where $\hat{b}_i = \frac{\ }{\alpha\|x_{t_0+}}$

Recall that we only need to stabilize this part.

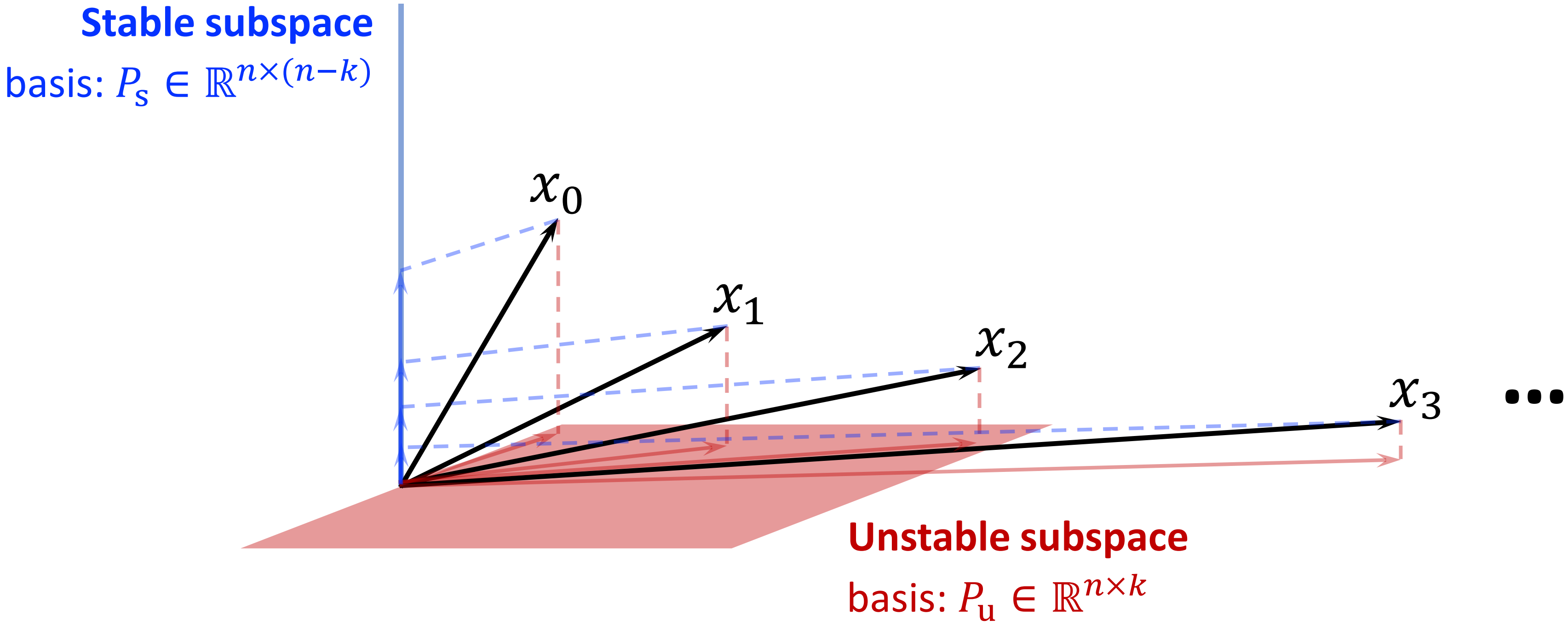
$$\begin{bmatrix} y_{\mathrm{u},t+1} \\ y_{\mathrm{s},t+1} \end{bmatrix} = \begin{bmatrix} \boxed{M_{\mathrm{u}} - P_{\mathrm{u}}^{\top} B K_{\mathrm{u}}} & 0 \\ -P_{\mathrm{s}}^{\top} B K_{\mathrm{u}} & M_{\mathrm{s}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix}$$

Stage 4: design the stabilizing controller

Return state feedback controller $u = -\widehat{K} x$, where $\widehat{K} = -\widehat{B}_{\mathrm{u}}^{-1} \widehat{M}_{\mathrm{u}} \widehat{P}_{\mathrm{u}}^{\top}$

Summary of our approach

- **Ingredient 1: subspace decomposition**
  - We show only info about a k-dim subspace is needed for stabilization.
- **Ingredient 2: subspace learning**
  - We design an algorithm using $O(k \log n)$ samples to learn the subspace.



Stability Guarantee (Orthogonal Case)

Main Result *(orthogonal case, informal version)*

For systems with *orthogonal* stable and unstable subspac
controllability assumption and other *regularity* as
approach can stabilize the system with number o

$$O(k \log n).$$

Incuring a state norm of $2^{O(k \log n)}$
Much smaller than worst-case $2^{\Theta(n)}$ when $k \ll n$.

- The big-O notation hides dependence on the following quantities:
    - $\log(\max(\|A\|, \|B\|))$
    - $\frac{1}{\log\frac{|\lambda_k|}{|\lambda_{k+1}|}}$: larger if the gap between unstable and stable eigenvalues is smaller.
    - $\log\frac{1}{c}$: larger when the unstable subspace is less controllable (c is a *controllability coefficient*).

Stability Guarantee (Orthogonal Case)

Main Result ***(orthogonal case, informal version)***

For systems with *orthogonal* stable and unstable subspaces, under proper *controllability* assumption and other *regularity* assumptions, the proposed approach can stabilize the system with number of samples less than

$$O(k \log n).$$

- Controllability assumption : $\sigma_{\mathrm{min}}(P_{\mathrm{u}}^{\top} B) \geq c||B||$ for some $c > 0$.
    - Can be relaxed to the usual strong controllability assumption $\sigma_{min}(\mathcal{C}) \geq \sigma$, where $\mathcal{C}$ is the controllability matrix.
    - But this comes at the cost of a worse sample complexity $O(k^2 \log n)$.

- Work in progress: relax the controllability assumption, yet keep the $O(k \log n)$ complexity.

Stability Guarantee (Orthogonal Case)

Main Result *(orthogonal case, informal version)*

For systems with *orthogonal* stable and unstable subspaces, under proper *controllability* assumption and other *regularity* assumptions, the proposed approach can stabilize the system with number of samples less than

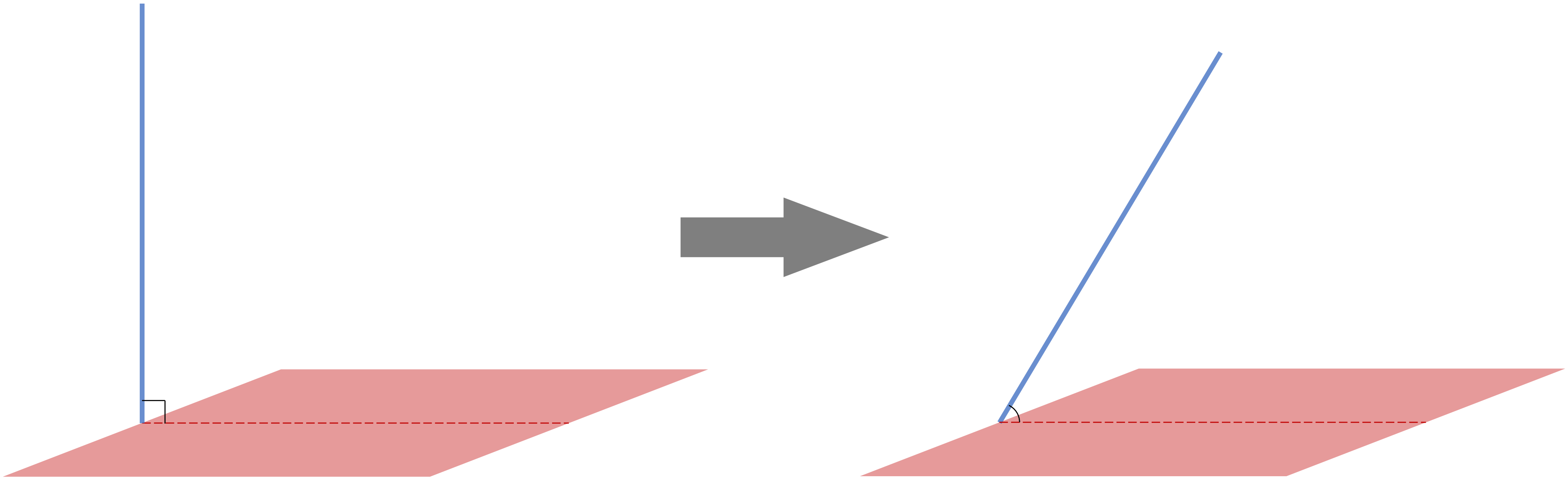
$$O(k \log n).$$

- Regularity assumptions:
    - A is diagonalizable, and has distinct eigenvalues that are not marginally stable.
    - The initial state is sampled uniformly at random from the unit sphere.

- Work in progress: the diagonalizable and distinct eigenvalue assumptions can be relaxed, but marginally stable eigenvalue is trickier to handle.

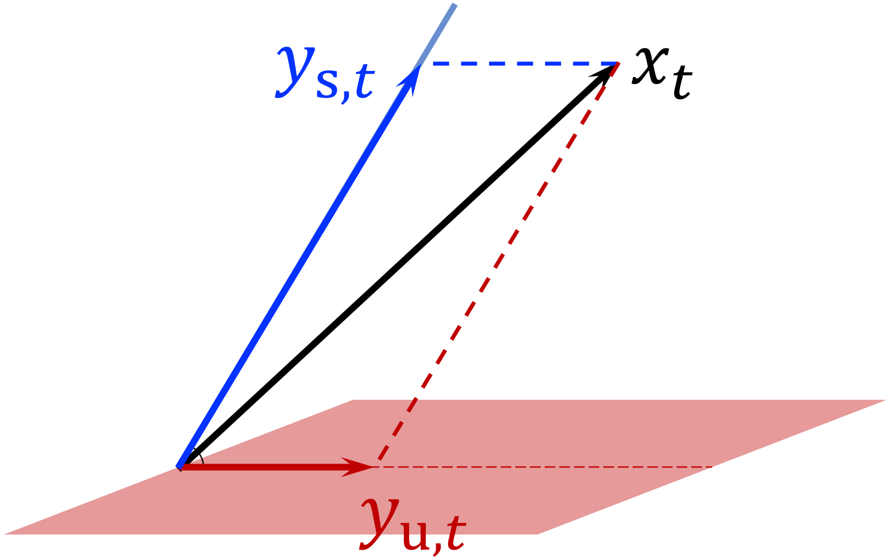
Generalize to Non-orthogonal Case

- We have achieved a $O(k \log n)$ sample complexity and avoided the norm to exponentially blow up in n when subspaces are *orthogonal*.
- How to generalize to the general *non-orthogonal* case?



Generalize to Non-orthogonal Case

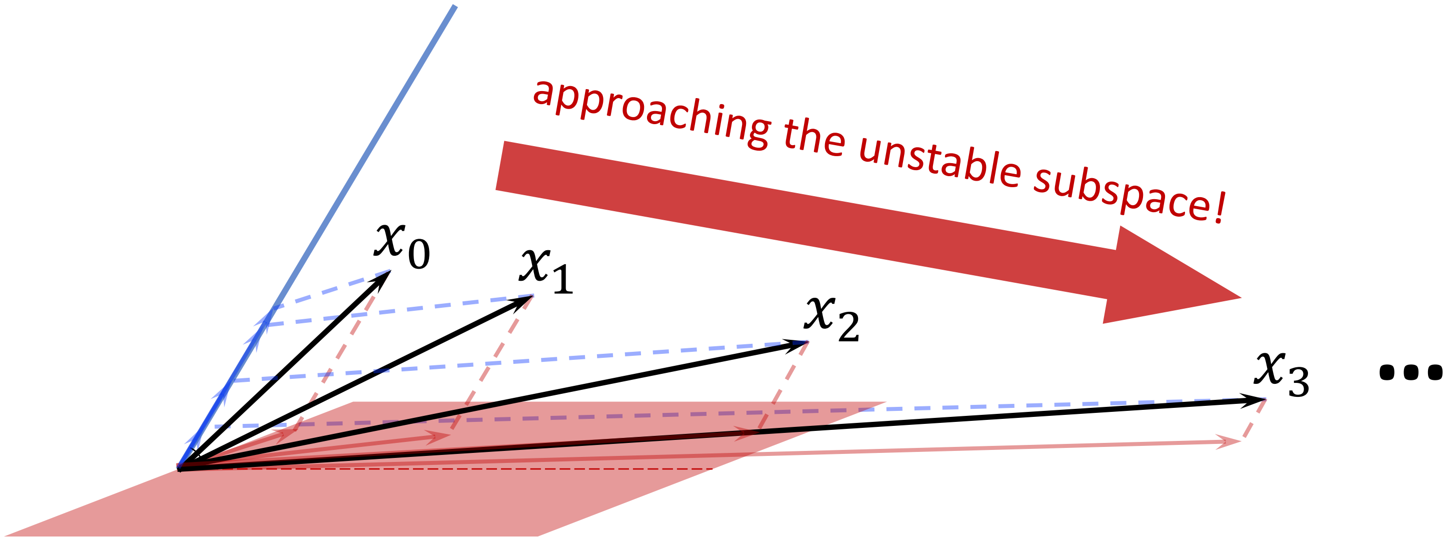
- **Main Challenge**: knowing $P_{\mathrm{u}}^{\top}$ alone is not enough for an *oblique* projection.



The unstable component is **NOT** $P_{\mathrm{u}}^{\top} x_t$.

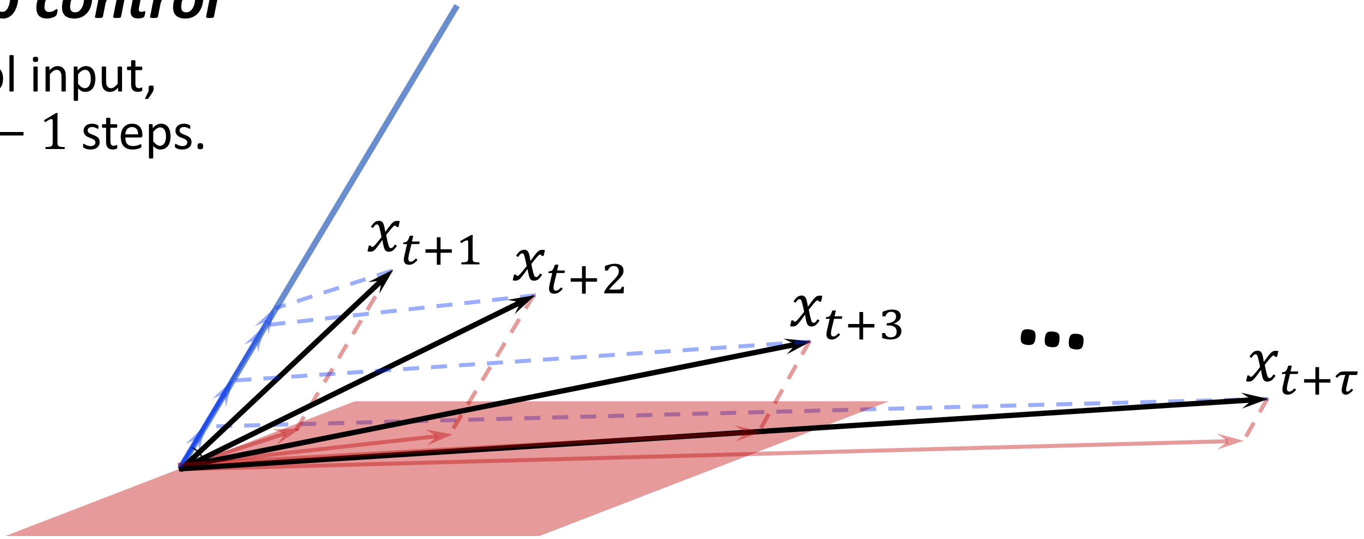
Generalize to Non-orthogonal Case

- **Main Challenge**: knowing $P_{\mathrm{u}}^{\top}$ alone is not enough for an *oblique* projection.
- **Idea:** let the system do the projection instead!
  - The open-loop system drives the state to unstable subspace after $\tau = O(1)$ steps.
  - No need to do decomposition anymore if $x_{t+\tau}$ is close to the unstable subspace.


approaching the unstable subspace!

Generalize to Non-orthogonal Case

- **Main Challenge**: knowing $P_{\mathrm{u}}^{\top}$ alone is not enough for an *oblique* projection.
- **Idea:** let the system do the projection instead!
  - The open-loop system drives the state to unstable subspace after $\tau = O(1)$ steps.
  - No need to do decomposition anymore if $x_{t+\tau}$ is close to the unstable subspace.
- **Control strategy: *τ-hop control***
  - After injecting a control input,
    run in open loop for $\tau - 1$ steps.



τ-hop Control

This block is non-zero due to non-orthogonality.

$$\begin{bmatrix} y_{\mathrm{u},t+1} \\ y_{2,t+1} \end{bmatrix} = \begin{bmatrix} M_{\mathrm{u}} & \Delta \\ 0 & M_2 \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{2,t} \end{bmatrix} + \begin{bmatrix} P_{\mathrm{u}}^\top B \\ P_2^\top B \end{bmatrix} u_t$$

We use subscript “2” because it is not exactly the stable subspace.

setting $u_t = -K_{\mathrm{u}} y_{\mathrm{u},t}$

and $u_{t+1} = \cdots = u_{t+\tau-1} = 0$

$$\begin{bmatrix} y_{\mathrm{u},t+\tau} \\ y_{2,t+\tau} \end{bmatrix} = \begin{bmatrix} M_{\mathrm{u}}^\tau - P_{\mathrm{u}}^\top A^{\tau-1} B K_{\mathrm{u}} & \Delta_\tau \\ -P_2^\top A^{\tau-1} B K_{\mathrm{u}} & M_2^\tau \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{2,t} \end{bmatrix}$$

τ-hop Control

can be made stable
via a properly designed K_{u}

$$\begin{bmatrix} y_{\mathrm{u},t+\tau} \\ y_{2,t+\tau} \end{bmatrix} = \begin{bmatrix} \boxed{M_{\mathrm{u}}^{\tau} - P_{\mathrm{u}}^{\top} A^{\tau-1} B K_{\mathrm{u}}} & \Delta_{\tau} \\ \boxed{-P_{2}^{\top} A^{\tau-1} B K_{\mathrm{u}}} & \boxed{M_{2}^{\tau}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t} \\ y_{2,t} \end{bmatrix}$$

approximately 0 since the
open-loop system does
the projection itself

already a stable matrix

Full Algorithm (General Case)

Stage 1: learning P_u

Let the system run open-loop for a period of time $t_0 = O(k \log n)$.

Set $\hat{P}_u$ as an orthonormal basis of the subspace spanned by $[x_{t_0+1}, \dots, x_{t_0+k}]$.

Stage 2: learning M_u

$$M_\mathrm{u} = \arg \min_{\hat{M}_\mathrm{u} \in \mathbb{R}^{k \times k}} \mathcal{L}\left(\hat{M}_\mathrm{u}\right) := \sum_{t=t_0+1}^{t_0+k} \left\| \hat{P}_\mathrm{u}^\top x_{t+1} - \hat{M}_\mathrm{u} \hat{P}_\mathrm{u}^\top x_t \right\|^2$$

Stage 3: learning $B_\tau = P_\mathrm{u}^\top A^{\tau-1} B$

For $i = 1, \dots, k$

    Let the system run in open loop for ω time steps.

    Run for τ more steps with initial $u_{t_i} = \alpha \left\| x_{t_i} \right\| e_i$, where $t_i = t_0 + k + i\omega + (i-1)\tau$.

Let $\hat{B}_\tau = \left[\hat{b}_1, \dots, \hat{b}_k\right]$ where $\hat{b}_i = \frac{1}{\alpha \| x_{t_i} \|} \left(\hat{P}_\mathrm{u}^\top x_{t_i+\tau} - \hat{M}_\mathrm{u}^\tau \hat{P}_\mathrm{u}^\top x_{t_i} \right)$.

Stage 4: design the stabilizing controller

Return state feedback controller $u = -\hat{K} x$, where $\hat{K} = -\hat{B}_\tau^{-1} \hat{M}_\mathrm{u}^\tau \hat{P}_\mathrm{u}^\top$

Stability Guarantee (General Case)

Main Result *(general case, informal version)*

For LTI systems under proper *controllability* assumption and other *regularity* assumptions, the proposed approach can stabilize the system with number of samples less than

$$O(k\log n).$$

- The big-O notation hides dependence on the following quantities:
    - $\log(\max(\|A\|,\|B\|))$
    - $\frac{1}{\log\frac{|\lambda_k|}{|\lambda_{k+1}|}}$: larger if the gap between unstable and stable eigenvalues is smaller.
    - $\log\frac{1}{c}$: larger when the unstable subspace is less controllable (c is a *controllability coefficient*).
    - $\log\frac{1}{1-\xi}$: $\xi := 1-\sigma_{\min}(P_s^\top P_2)$ measures *"degree of orthogonality"* of the stable/unstable subspaces ($\xi = 0$ for orthogonal, and $\xi = 1$ for linearly dependent).

Stability Guarantee (General Case)

Main Result *(general case, informal version)*

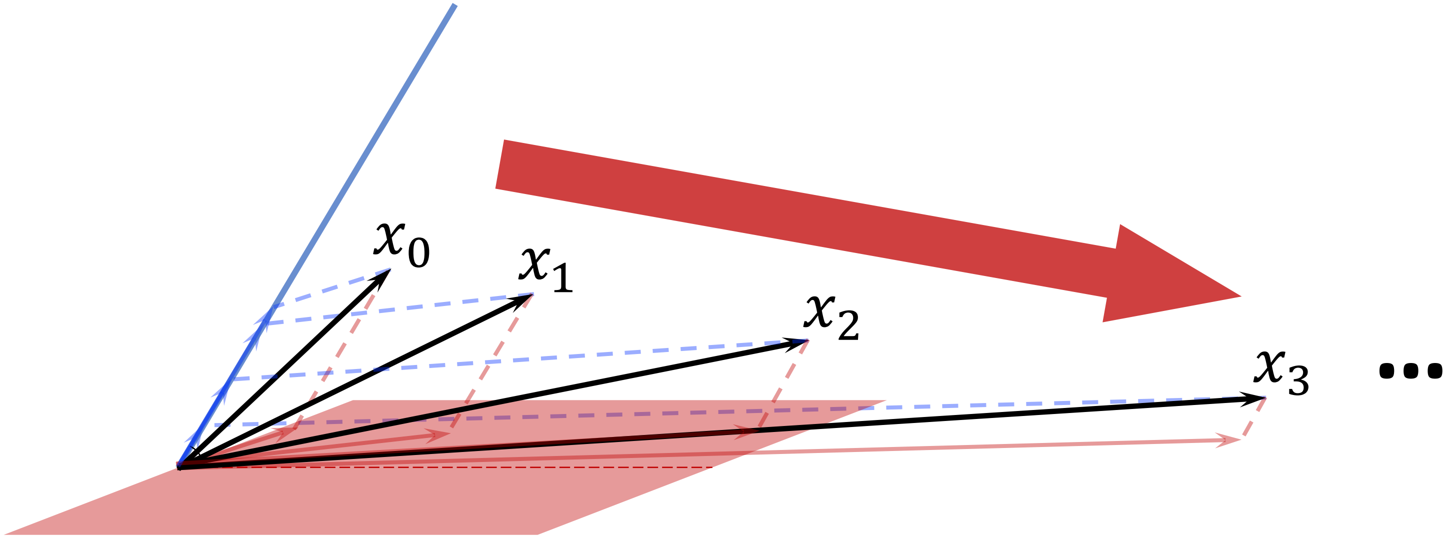
For LTI systems under proper *controllability* assumption and other *regularity assumptions*, the proposed approach can stabilize the system with number of samples less than

$$O(k \log n).$$

- Regularity assumptions:
  - A is diagonalizable, and has distinct eigenvalues that are not marginally stable.
  - The initial state is sampled uniformly at random from the unit sphere.
  - $\frac{|\lambda_1|^2|\lambda_{k+1}|}{|\lambda_k|}$ should be small (since we want the τ-hop dynamics to be stable).

How to Handle the Noisy Case?

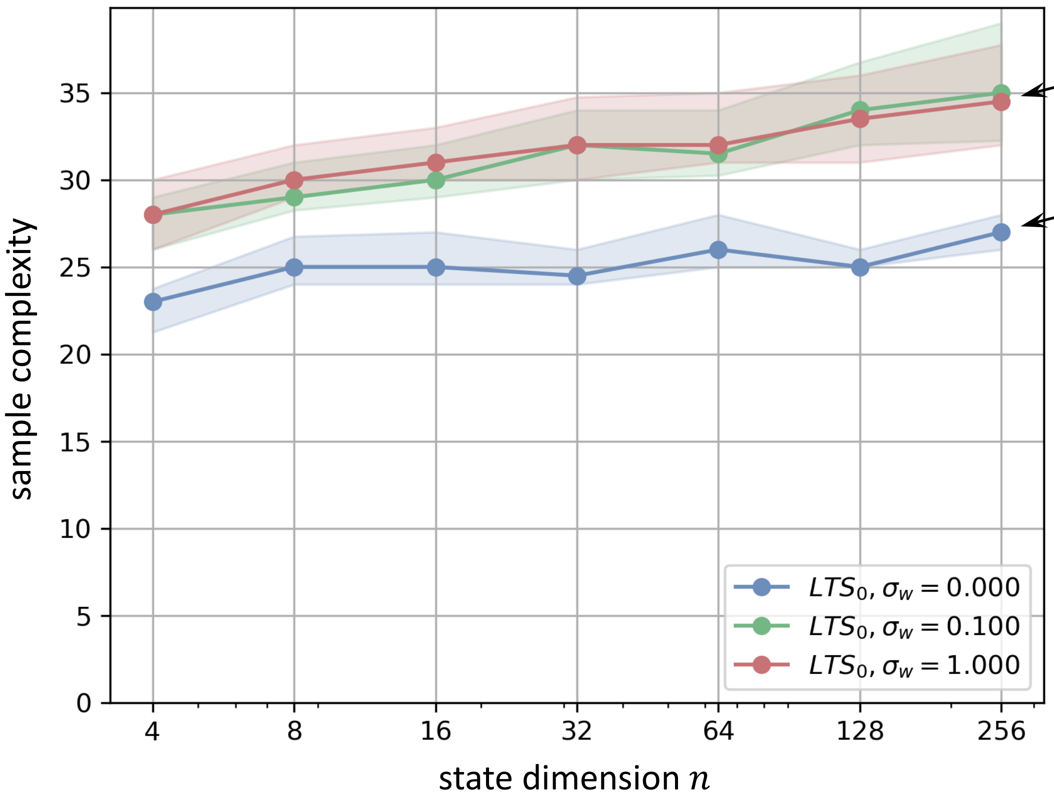
$$\begin{aligned}\begin{bmatrix} y_{\mathrm{u},t} \\ y_{\mathrm{s},t} \end{bmatrix} &= \begin{bmatrix} M_{\mathrm{u}} & 0 \\ 0 & M_{\mathrm{s}} \end{bmatrix} \begin{bmatrix} y_{\mathrm{u},t-1} \\ y_{\mathrm{s},t-1} \end{bmatrix} + w_{t-1} \\ &= \begin{bmatrix} M_{\mathrm{u}}^t y_{\mathrm{u},0} + M_{\mathrm{u}}^{t-1} w_{\mathrm{u},1} + \cdots + w_{\mathrm{u},t-1} \\ M_{\mathrm{s}}^t y_{\mathrm{s},0} + M_{\mathrm{s}}^{t-1} w_{\mathrm{s},1} + \cdots + w_{\mathrm{s},t-1} \end{bmatrix} \begin{matrix} \text{blow up} \\ \text{stay as } O(1) \end{matrix}\end{aligned}$$



The open-loop system still functions as automatic projection!

How to Handle the Noisy Case?

$k = 3, n$ varying

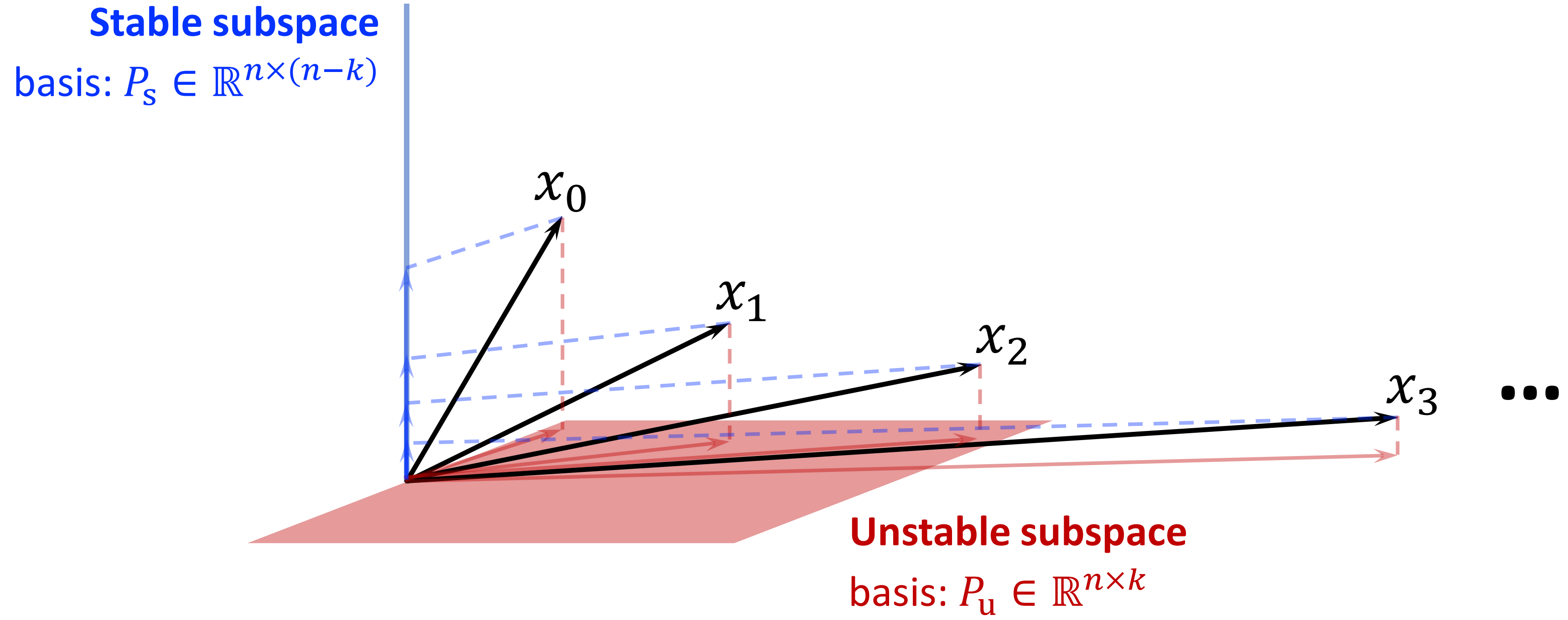


with Gaussian noise

without Gaussian noise

Major take-home messages

- Learning only partial information about (A, B) is necessary for stabilization.
  - Utilize the *stable/unstable subspace decomposition*.
- In this way a sample complexity of $O(k \log n)$ can be achieved.
  - A novel instance-specific bound for the learn-to-stabilize problem.
  - Avoid exponential state-norm blowup in n.



Stable subspace

basis: $P_{\mathrm{s}} \in \mathbb{R}^{n\times(n-k)}$

Unstable subspace

basis: $P_{\mathrm{u}} \in \mathbb{R}^{n\times k}$

Major take-home messages

- Learning only partial information about (A, B) is necessary for stabilization.
  - Utilize the *stable/unstable subspace decomposition*.
- In this way a sample complexity of $O(k \log n)$ can be achieved.
  - A novel instance-specific bound for the learn-to-stabilize problem.
  - Avoid exponential state-norm blowup in n.

Future directions:

- Analyze the noisy case.
- Relax regularity assumptions.
- Provide lower bounds.
- Consider output feedback systems.

Reference:

Yang Hu, Adam Wierman, Guannan Qu, “On the Sample Complexity of Stabilizing LTI Systems on a Single Trajectory”, accepted to NeurIPS 2022. arXiv preprint: arXiv:2202.07187

On the Sample Complexity of Stabilizing LTI Systems on a Single Trajectory

Yang Hu (Harvard), Adam Wierman (Caltech), Guannan Qu (CMU)

Presented by: Guannan Qu

Assistant Professor of ECE, Carnegie Mellon University

Sept 29, 2022