Stabilizing reinforcement learning control: A modular framework for optimizing over all stable behavior*

Nathan P. Lawrence[a], Philip D. Loewen[a], Shuyuan Wang[b], Michael G. Forbes[c], R. Bhushan Gopaluni[b]

[a]*Department of Mathematics, University of British Columbia, Vancouver BC, Canada*
[b]*Department of Chemical and Biological Engineering, University of British Columbia, Vancouver, BC Canada*
[c]*Honeywell Process Solutions, North Vancouver, BC Canada*

Abstract

We propose a framework for the design of feedback controllers that combines the optimization-driven and model-free advantages of deep reinforcement learning with the stability guarantees provided by using the Youla-Kučera parameterization to define the search domain. Recent advances in behavioral systems allow us to construct a data-driven internal model; this enables an alternative realization of the Youla-Kučera parameterization based entirely on input-output exploration data. Perhaps of independent interest, we formulate and analyze the stability of such data-driven models in the presence of noise. The Youla-Kučera approach requires a stable "parameter" for controller design. For the training of reinforcement learning agents, the set of all stable linear operators is given explicitly through a matrix factorization approach. Moreover, a nonlinear extension is given using a neural network to express a parameterized set of stable operators, which enables seamless integration with standard deep learning libraries. Finally, we show how these ideas can also be applied to tune fixed-structure controllers.

Keywords: Reinforcement learning, data-driven control, Youla-Kučera parameterization, neural networks, stability, process control

1. Introduction

Closed-loop stability is a basic requirement in controller design. However, many learning-based control schemes do not address it explicitly [1]. This is somewhat understandable. First, the "model-free" setup assumed in such algorithms, compounded by the complexity of the methods and their underlying data structures, makes stability difficult to reason about. Second, especially in the case of reinforcement learning (RL), many of the striking recent success stories pertain to simulated tasks or game-playing environments in which catastrophic failure has no real-world impact. When a feedback controller is to be learned directly with RL, system stability during exploration (along with learning performance) is influenced by the discount factor, reward function, and numerous other hyperparameters [1]. Figure 1 illustrates this point. These issues provide a counterpoint to the generality and expressive capacity of modern RL algorithms, which have nonetheless attracted immense interest for control tasks [2].

1.1. Contributions

We propose a *stability-preserving framework* for RL-based controller design. Our inspiration is the Youla-Kučera (YK) parameterization [3], which characterizes all stabilizing controllers for a given system. We formulate a "model-free" realization of the YK parameterization from exploration data, enabling an RL agent to optimize over all stable closed-loop behavior in an unconstrained fashion. Specifically, we leverage tools from the behavioral systems literature [4]: a Hankel matrix of input-output data serves as an internal model through a dynamic variation of Willems'

*This paper has been accepted for publication in Automatica. The most up-to-date version is available here: arXiv:2310.14098.

Email addresses: input@nplawrence.com (Nathan P. Lawrence), loew@math.ubc.ca (Philip D. Loewen), bhushan.gopaluni@ubc.ca (R. Bhushan Gopaluni)

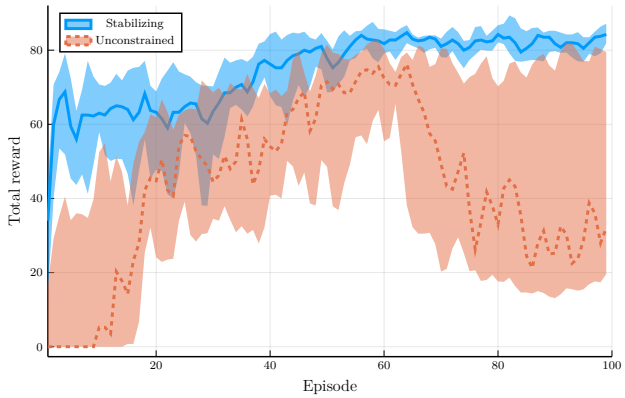

Figure 1: Cumulative reward over two PI-tuning experiments: one using the proposed stabilizing framework and the other using standard RL. The stability-agnostic agent often destabilizes the system and struggles to recover.

fundamental lemma. Under this regime, an RL agent is able to directly manipulate the closed-loop dynamics through a learnable stable operator. We show how this stable operator can be deployed in an unconstrained and seamless fashion for both linear and nonlinear control strategies.

Perhaps of independent interest, we formulate a data-driven stability criterion in terms of the Hankel matrix structure commonly used in data-driven control. Output noise complicates the situation when working with Hankel matrices. We provide probabilistic analysis for the stability of such models as well.

In sum, we disentangle three key components of RL-based control system design: Algorithms, function approximators, and dynamic models. Moreover, our framework supports a modular approach to learning stabilizing policies, in which advances in any single category can be applied to improve overall results.

1.2. Related work

Buşoniu et al. [1] provide a survey of RL techniques from a control-theoretic perspective, emphasizing the need for stability-aware RL algorithms. Since one of the appeals of RL is model-free policy optimization, methods for incorporating stability vary widely based on prior assumptions about the underlying dynamics. As such, a wide variety of approaches have been proposed. Relatively early methods for incorporating stability into RL are based on integral quadratic constraints (IQCs) to capture nonlinearities or time-varying components in the environment or policy structure [5, 6]. In the context of RL, nonlinearities in the environment or the nonlinear activation functions used to construct a policy neural network can be characterized using IQCs. This is also the basis for more recent approaches [7, 8, 9, 10]. Lyapunov theory is another popular framework in the RL literature [11, 12, 13, 14, 15, 16, 17]. The principal idea is to learn a policy that satisfies the decrease condition for a suitable Lyapunov function. Similarly, the linear quadratic regulator is a fruitful testbed for benchmarking and analyzing RL algorithms; several works develop stability guarantees when the system dynamics are not available to the RL agent [18, 19, 20].

The YK parameterization is seemingly an under-utilized technique for incorporating stability into RL algorithms, with some examples due to Roberts et al. [21], Friedrich and Buss [22]. Roberts et al. [21] propose its use after evaluating the performance of RL with several different controller parameterizations for a simulated ball-catching task. Subsequently, Friedrich and Buss [22] employ the YK parameterization through the use of a crude plant model; RL is used to optimize the tracking performance of a physical two degrees of freedom robot in a safe fashion while accounting for unmodeled nonlinearities. Recently, a recurrent neural network architecture based on IQCs was developed [8]. Since this architecture satisfies stability conditions by design, it can be used for control in a nonlinear version of the YK parameterization [10].

While we also use the YK parameterization, our approach has several novel aspects. We propose to produce stable operators using a non-recurrent neural network structure; this makes the implementation and integration with off-the-shelf RL algorithms relatively straightforward, for both on-policy and off-policy learning. This contrasts with IQC or Lyapunov-based approaches, such as Jin and Lavaei [7], Zhang et al. [11], Modares et al. [12], that place strong structural hypotheses on the network architectures and update schemes. We also formulate a data-driven realization of

the YK parameterization based on Willems' fundamental lemma, essentially removing the need for prior modeling, whereas the cited works based on the YK parameterization assume that a parameterized model is given. Moreover, we establish the stability of such data-based models, a principal hypothesis in the YK parameterization. Finally, we show how the techniques presented here can be applied to a fixed-structure controller, an aspect not covered in other YK-based approaches.

1.3. Notation

Given a matrix $M \in \mathbb{R}^{m\times n}$, we write $\|M\|_F = \left(\sum_{i,j} m_{i,j}^2\right)^{\frac{1}{2}}$ for the Frobenius norm and $\|M\|$ for the spectral norm, that is, the largest singular value. M^+ denotes the Moore-Penrose pseudoinverse. (Often $m \le n$ and M has full rank, in which case $M^+ = M^\top \left(MM^\top\right)^{-1}$.) When $m = n$, we indicate the spectral radius by $\rho(M) = \max\{|\lambda|\colon \lambda \text{ is an eigenvalue of } M\}$. If $M = M^\top$, we write $M > 0$ (or $M \ge 0$) instead of saying M is positive-definite (resp. semi-definite).

2. Background

We consider a nominal linear time-invariant (LTI) system whose state x evolves in $\mathbb{R}^n$:

$$P\begin{cases} x_{t+1} = Ax_t + Bu_t \\ \quad y_t = Cx_t, \qquad\qquad t = 0, 1, 2, \dots. \end{cases} \tag{1}$$

The corresponding transfer function is $P(z) = C(zI - A)^{-1}B$. We treat the constant matrices A, B, C as unknown, and lay the foundation for Willems' fundamental lemma and the YK parameterization with the following mild assumptions.

Assumption 2.1. *An upper bound of the state dimension n is available.*

Assumption 2.2. *The matrix pair (A, B) is controllable, and the pair (A, C) is observable.*

Assumption 2.3. *The nominal system is stable and single-input, single-output (SISO), that is, $\rho(A) < 1$, $B \in \mathbb{R}^{n\times 1}$, and $C \in \mathbb{R}^{1\times n}$.*

2.1. A dynamic Willems' lemma as an internal model

Given an N-element sequence $\{z_t\}_{t=0}^{N-1}$ of vectors in $\mathbb{R}^m$ and an integer L, $1 \le L \le N$, the *Hankel matrix of order L* is the $mL \times (N - L + 1)$ array with the constant skew-diagonal structure

$$H_L(z) = \begin{bmatrix} z_0 & z_1 & \cdots & z_{N-L} \\ z_1 & z_2 & \cdots & z_{N-L+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_{L-1} & z_L & \cdots & z_{N-1} \end{bmatrix}.$$

Situations where this matrix has linearly independent rows are of particular interest.

Definition 2.4. *The sequence $\{z_t\}_{t=0}^{N-1} \subset \mathbb{R}^m$ is* persistently exciting of order L *if* $\operatorname{rank}(H_L(z)) = mL$.

Definition 2.5. *An input-output sequence $\{u_t, y_t\}_{t=0}^{N-1}$ is a* trajectory *of an LTI system (A, B, C) if there exists a state sequence $\{x_t\}_{t=0}^{N-1}$ such that Eq. (1) holds.*

The following theorem is the state-space version of Willems' fundamental lemma [23, 4]. It provides an alternative characterization of an LTI system based entirely on input-output data.

Theorem 2.6 (See van Waarde et al. [24])**.** *Let $\{u_t, y_t\}_{t=0}^{N-1}$ be a trajectory of an LTI system (A, B, C) where u is persistently exciting of order $L + n$. Then $\{\overline{u}_t, \overline{y}_t\}_{t=0}^{L-1}$ is a trajectory of (A, B, C) if and only if there exists $\alpha \in \mathbb{R}^{N-L+1}$ such that*

$$\begin{bmatrix} H_L(u) \\ H_L(y) \end{bmatrix} \alpha = \begin{bmatrix} \overline{u} \\ \overline{y} \end{bmatrix}. \tag{2}$$

Here the right-hand side is the block-structured column vector formed from $\overline{u} = [u_0 \dots u_{L-1}]^\top$ and $\overline{y} = [y_0 \dots y_{L-1}]^\top$.

In applications, one uses measured input-output data to construct the left-hand side in Eq. (2). Then, to test whether a candidate input-output sequence of length L is indeed a system trajectory, one uses it as the right-hand side and attempts to solve for α [4, 24, 25].

We now formulate a dynamic variant of Theorem 2.6, enabling one to advance a trajectory in time. Given N and vectors $z_0, \ldots, z_N$, let $z = \{z_t\}_{t=0}^{N-1}$ and $z' = \{z_t\}_{t=1}^{N}$. Then let

$$H'_L(z) = H_L(z').$$

Note that $H'_L(z)$ has the same shape as $H_L(z)$.

Given a system trajectory $\{u_t, y_t\}_{t=0}^{L-1}$ on the right-hand side of Eq. (2) with $L \geq n$, we note that the next output y_L is uniquely determined by these available data. Intuitively, a time-shifted Hankel matrix advances the internal, unknown state of the system forward resulting in y_L.

Corollary 2.7. *Let $\{u_t, y_t\}_{t=0}^{N}$ be a trajectory of a strictly proper LTI system (A, B, C) where u is persistently exciting of order $L + 1 + n$. Then for each trajectory $\{\overline{u}_t, \overline{y}_t\}_{t=0}^{L-1}$ of (A, B, C), there exists $\alpha \in \mathbb{R}^{N-L+1}$ such that*

$$\overline{y}' = H'_L(y)\alpha. \tag{3}$$

Remark 2.8. *The hypotheses are to ensure both $\begin{bmatrix} H_L(u) \\ H_L(y) \end{bmatrix}$ and $\begin{bmatrix} H'_L(u) \\ H'_L(y) \end{bmatrix}$ satisfy the requirements in Theorem 2.6.* ◁

Proof. By Theorem 2.6, the trajectory $\{\overline{u}_t, \overline{y}_t\}_{t=0}^{L-1}$ satisfies

$$\begin{bmatrix} H_L(u) \\ H_L(y) \end{bmatrix} \alpha = \begin{bmatrix} \overline{u} \\ \overline{y} \end{bmatrix}$$

for some $\alpha \in \mathbb{R}^{N-L+1}$. Moreover, by Definition 2.5 there exists a sequence of states $\{\overline{x}_t\}_{t=0}^{L-1}$ that corresponds to the input-output trajectory $\{\overline{u}_t, \overline{y}_t\}_{t=0}^{L-1}$. This sequence induces the state $\overline{x}_L$. We have

$$\begin{aligned}
\sum_{i=0}^{N-L} \alpha_i y_{L+i} &= \sum_{i=0}^{N-L} \alpha_i C \left(A x_{L-1+i} + B u_{L-1+i}\right) \\
&= C\left(A \sum_{i=0}^{N-L} \alpha_i x_{L-1+i} + B \sum_{i=0}^{N-L} \alpha_i u_{L-1+i}\right) \\
&= C\left(A\overline{x}_{L-1} + B\overline{u}_{L-1}\right) \\
&= C\overline{x}_L \\
&= \overline{y}_L
\end{aligned}$$

as desired. □

Algorithm 1 shows how to use this scheme in Corollary 2.7 for closed-loop simulation. Moreover, this idea is particularly useful for aligning the true system with an internal Hankel representation.

Algorithm 1: Data-driven simulation

Input: Data $\{u_k, y_k\}_{k=0}^{N}$ with persistently exciting input of order $L + 1 + n$; Initial trajectory $\{\overline{u}_k, \overline{y}_k\}_{k=0}^{L-1}$

for each time step **do**
- Solve for α: $\begin{bmatrix} H_L(u) \\ H_L(y) \end{bmatrix} \alpha = \begin{bmatrix} \overline{u} \\ \overline{y} \end{bmatrix}$
- Compute the next element $\overline{y}' = H'_L(y)\alpha$
- Generate the next control input $\overline{u}_L$
- Update trajectory: $\{\overline{u}_k, \overline{y}_k\}_{k=0}^{L-1} \leftarrow \{\overline{u}_k, \overline{y}_k\}_{k=1}^{L}$

2.2. Data-driven realization of the Youla-Kučera parameterization

We consider the standard four sensitivity functions associated with a plant P and controller K: $\frac{PK}{1+PK}, \frac{P}{1+PK}, \frac{K}{1+PK}, \frac{1}{1+PK}$. The YK parameterization produces the set of all stabilizing controllers through a combination of an internal system model and a stable operator. The trick is to parameterize the aforementioned closed-loop transfer functions, then recover a controller. For example, the response of the transfer function $\frac{PK}{1+PK}$ from the reference r to output y is determined by the transfer function $\frac{K}{1+PK}$. By introducing a stable design variable Q, we can then directly shape the stable behavior of the system through the transfer function PQ. By asserting $Q = \frac{K}{1+PK}$ and solving for K, we arrive at the YK parameterization [3]:

$$\mathcal{K}_{\text{stable}} = \left\{ \frac{Q}{1 - QP} : Q \text{ is stable} \right\}. \tag{4}$$

Indeed, for a stable plant P, all four sensitivity functions are stable for any K in $\mathcal{K}_{\text{stable}}$. Moreover, when P is linear, one may use a nonlinear operator Q to parameterize nonlinear controllers [3, 10].

In Algorithm 2, we translate the mathematical ideas above into a direct sequential process. In particular, we utilize Corollary 2.7 in conjunction with the feedback connections in Eq. (4) to produce stabilizing actions. Theorem 2.10 provides details of the correspondence.

Algorithm 2: Data-driven stabilizing controller

Input: Stable parameter Q; Data $\{u_k, y_k\}_{k=0}^{N}$ with persistently exciting input of order $L + 1 + n$; Initial trajectory $\{\overline{u}_k, \overline{y}_k\}_{k=0}^{L-1}$

1 **for** each time step t **do**
2 Set $u_{t-1} \leftarrow \overline{u}_{L-1}$
3 Observe the tracking error $e_t = r_t - y_t$ from the system
4 Compute $\overline{y}_L$ from Eq. (3)
5 Apply the input $\widehat{r} = e_t + \overline{y}_L$ to the Q parameter and return control action $\overline{u}_L$; for example, step forward in time of an LTI representation of Q
6 Update the trajectory: $\{\overline{u}_k, \overline{y}_k\}_{k=0}^{L-1} \leftarrow \{\overline{u}_k, \overline{y}_k\}_{k=1}^{L}$

Remark 2.9. *Notice in Line 5 that Q ideally parameterizes the input-output dynamics between the reference r and controls u. In practice, Q takes into account discrepancies between the true plant output y_t (Line 3) and the internal prediction $\overline{y}_L$ (Line 4).* ◁

Theorem 2.10. *Assume P is a stable and strictly proper LTI system. Let Q be a stable and proper LTI parameter. Given an upper bound L of the order of P, Algorithm 2 produces the same control signal $\{\overline{u}_t\}_{t=0}^{\infty}$ as the YK parameterization.*

Proof. We use q_t, p_t to denote the impulse responses of Q and P, respectively. Similarly, respective minimal state-space matrices are denoted (A_q, B_q, C_q, D_q) and (A_p, B_p, C_p).

By the YK parameterization, we have $U = KE$ for the controller $K \in \mathcal{K}_{\text{stable}}$ given by

$$\begin{aligned}
& & K(z) &= \frac{Q(z)}{1 - Q(z)P(z)} \quad \forall z \in \mathbb{C} \\
\iff & & (1 - Q(z)P(z))\, U(z) &= Q(z)E(z) \\
\iff & & u_t &= q_t * (e_t + p_t * u_t) \quad \forall t \in \mathbb{N}_0 \\
& & &= \sum_{j=0}^{t-1} C_q A_q^{t-1-j} B_q \widehat{r}_j + D_q \widehat{r}_t,
\end{aligned} \tag{5}$$

where $\widehat{r}_j = e_j + \sum_{i=0}^{j-1} C_p A_p^{j-1-i} B_p u_i$ and $*$ is the convolution operator; we have also assumed, without loss of generality, that P and Q have zero initial state.

Next we relate Eq. (5) to Algorithm 2. Let $\{e_k\}_{k=0}^{\infty}$ be an arbitrary sequence. (Such a sequence is dynamically generated in Algorithm 2.) Without loss of generality, let the initial trajectory be $\{\overline{u}_k, \overline{y}_k\}_{k=0}^{L-1} = \{0, 0\}_{k=0}^{L-1}$. For each time

$t \in \mathbb{N}_0$ we compute $\alpha^{(t)}$ and $\bar{y}_t = \bar{y}_L$ from Eq. (3). Since L is an upper bound on the order of P, $\bar{y}_t$ is the unique next output from the trajectory $\{\bar{u}_k, \bar{y}_k\}_{k=0}^{L-1}$. Therefore, we have $\widehat{r}_t = e_t + \sum_{i=0}^{N-L} \alpha_i^{(t)} y_{L+i}$. Then $\bar{u}_t = \sum_{j=0}^{t-1} C_q A_q^{t-1-j} B_q \widehat{r}_j + D_q \widehat{r}_t$ gives the next control input.

By updating the trajectory between time steps—$\{\bar{u}_k, \bar{y}_k\}_{k=0}^{L-1} \leftarrow \{\bar{u}_k, \bar{y}_k\}_{k=1}^{L}$—we dynamically generate a sequence $\{\alpha^{(t)}\}_{t=0}^{\infty}$ that produces the control inputs $\{\bar{u}_t\}_{t=0}^{\infty}$ satisfying the discrete integral equation in Eq. (5). □

3. On the stability of noisy Hankel matrices

Theorem 2.10 assumes the underlying system is open-loop stable, in which case one may utilize Algorithm 2 to produce stabilizing control actions. However, the long-term predictions generated by Corollary 2.7 will be influenced by the noise in the data and singular values of the resulting stacked Hankel matrices. Stopping the data collection process early can result in unstable predictions even for an open-loop stable plant; see the initial spectral radius values in Fig. 2.

3.1. Data-driven stability test

We formulate the stability of a Hankel matrix system representation by delving deeper into the recursive nature of infinite trajectories generated by Algorithm 1. This involves deriving a special matrix structure that relates successive solutions $\alpha_t, \alpha_{t+1}, \ldots$ from Eq. (2).

Start with the equation from Theorem 2.6: let α_0 denote the minimum-norm solution of

$$\begin{bmatrix} H_L(u) \\ H_L(y) \end{bmatrix} \alpha_0 = \begin{bmatrix} \bar{u} \\ \bar{y} \end{bmatrix}.$$

(Henceforth we assume minimum-norm solutions; any solution may be used, but the minimum-norm solution will lead to a clean formulation.) By Corollary 2.7 we then have that the successive output trajectory is given by $\bar{y}' = H'(y)\alpha_0$. By extension, the next trajectory is given by $\begin{bmatrix} \bar{u}' \\ \bar{y}' \end{bmatrix} = \begin{bmatrix} H'_L(u) \\ H'_L(y) \end{bmatrix} \alpha_0$.

Starting from $\begin{bmatrix} \bar{u}' \\ \bar{y}' \end{bmatrix}$, we repeat the process to arrive at the recursion

$$\underbrace{\begin{bmatrix} H_L(u) \\ H_L(y) \end{bmatrix}}_{H} \alpha_{t+1} = \underbrace{\begin{bmatrix} H'_L(u) \\ H'_L(y) \end{bmatrix}}_{H'} \alpha_t, \qquad t = 0, 1, 2, \ldots. \tag{6}$$

This can be seen as the "free response" of the α_t dynamics inferred from the collected data and produced by Algorithm 1.[1] Therefore, by checking the eigenvalues of H^+H' we determine if the matrix transformation from a system's Hankel matrix to its time-shifted counterpart is internally "contractive"; under the assumption of minimum-norm solutions, this implies that the behavior $\begin{bmatrix} \bar{u} \\ \bar{y} \end{bmatrix}$ is bounded and $\begin{bmatrix} \bar{u} \\ \bar{y} \end{bmatrix} \to 0$ as $t \to \infty$. In the ensuing sections, we formalize and prove properties about this special matrix structure in the presence of noise.

3.2. Random Hankel matrices

Randomness complicates the notion of stability. Going forward, we assume the outputs have the form $y_t + \omega_t$ where ω_t is normally distributed. In order to characterize the eigenvalues of H^+H' under measurement noise, we first isolate the underlying random Hankel matrix in the term $H_L(y) + H_L(\omega)$. We will then be able to relate properties of the random matrix $H_L(\omega)$ to the overall structure. Therefore, this subsection focuses on Hankel matrices of purely random signals as $N \to \infty$, and then the latter section re-introduces the input-output dynamics.

Given a sequence of independent random variables $\omega_0, \omega_1, \ldots$, we consider the doubly-infinite array

$$\mathcal{H} = \begin{bmatrix} \omega_0 & \omega_1 & \omega_2 & \cdots \\ \omega_1 & \omega_2 & \omega_3 & \cdots \\ \omega_2 & \omega_3 & \omega_4 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{7}$$

[1] To evaluate the free response of the dynamics in Eq. (1) in Hankel form, one may add the constraint $H'_L(u)\alpha_t = 0$ to Eq. (6).

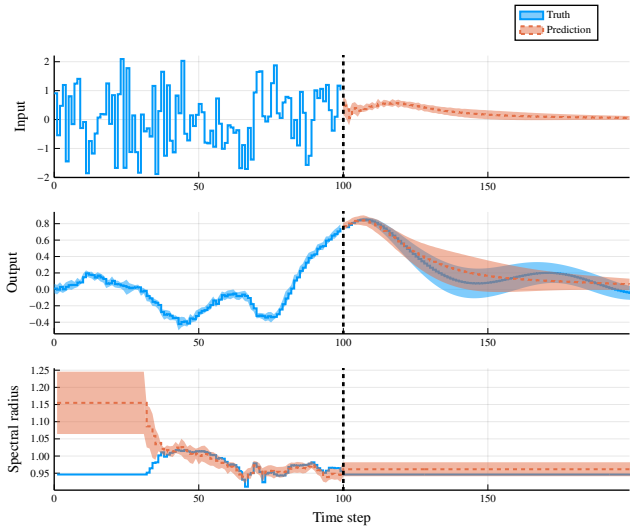

Figure 2: 100 time steps of input-output data are collected using a standard normal probing signal. The recursion in Eq. (6) is used to continue the rollout. This is done several times for different samples of output noise. The bottom figure is the evolution of the spectral radii for the noisy and noise-free matrices H^+H'.

Our focus is on short wide submatrices anchored at the top left corner. Specifically, for any fixed positive integers L and N, we write $H_{L,N}$ for the top left $L \times N$ submatrix of $\mathcal{H}$.

Equation (8) is a fundamental concentration inequality that drives our analysis of random Hankel matrices. Corollary 3.2 is a useful special case.

Lemma 3.1 (Hanson-Wright inequality, adapted from [26])**.** *There exists a constant $c > 0$ such that, for every $n \times n$ matrix M, any random vector $X = (X_0, \ldots, X_{n-1}) \in \mathbb{R}^n$ with independent standard normal components X_i obeys*

$$\mathbb{P}\left\{\left|X^\top MX - \mathbb{E}\left[X^\top MX\right]\right| > t\right\} \leq 2\exp\left(-c\min\left\{\frac{t^2}{\|M\|_F^2}, \frac{t}{\|M\|}\right\}\right), \qquad t \geq 0. \tag{8}$$

Corollary 3.2. *Let $X_0, X_1, \ldots$ be a sequence of standard normal random variables. Then there exist constants $c_0, c_1 > 0$ such that for any $n \in \mathbb{N}$ and any $\alpha \in (0,1)$, one has both*

(a) $\mathbb{P}\left\{\left|\sum_{k=0}^{n-1} X_k^2 - n\right| < \alpha n\right\} \geq 1 - 2\exp\left(-c_0\alpha^2 n\right)$, *and*

(b) $\mathbb{P}\left\{\left|\sum_{k=0}^{n-1} X_k X_{\sigma(k)}\right| < \alpha n\right\} \geq 1 - 2\exp\left(-c_1\alpha^2 n\right)$,
for any $\sigma\colon \mathbb{N} \to \mathbb{N}$ such that $\sigma(k) \neq k$ for all k.

Proof. Both parts follow from taking $t = \alpha n$ in Eq. (8), and using $0 < \alpha < 1$ to simplify $\min\left\{\alpha^2 n, \alpha n\right\} = \alpha^2 n$. In part (a), one uses the $n \times n$ identity matrix for M: clearly $\|I\|_F^2 = n$ and $\|I\| = 1$. In part (b), one defines $K = \max\{\sigma(1), \ldots, \sigma(n)\}$ and forms M as a $K \times K$ matrix in which every entry is 0 except for the n entries at positions $(k, \sigma(k))$, each of which equals 1. Again $\|M\|_F^2 = n$ and $\|M\| = 1$. □

Proposition 3.3 analyzes the limiting behavior of the singular values of random Hankel matrices as the number of samples tends to infinity. Its proof, shown in Appendix B, is a key step toward a spectral analysis of an interesting combination of related matrices.

Proposition 3.3. *Suppose each of the independent random variables ω_i has a standard normal distribution. Then for each fixed $L \geq 1$, there is a sequence $r_0, r_1, \ldots$, with $r_N \to \infty$ as $N \to \infty$, such that*

$$\lim_{N\to\infty} \mathbb{P}\left\{\sigma_{\min}\left(H_{L,N}\right) > r_N\right\} = 1.$$

Here $\sigma_{\min}(\cdot)$ returns the smallest singular value of its matrix argument.

Recall the matrix $\mathcal{H}$ in Eq. (7); for any fixed positive integers L and N, we consider the top left submatrix of shape $L \times (N+1)$, namely, $\overline{H}_{L,N} = H_{L,N+1}$, and extract two $L \times N$ chunks of interest. These are the "standard" Hankel matrix $H = H_{L,N}$ discussed above, formed by removing the last column, and the "time-shifted" Hankel matrix $H' = H'_{L,N}$, formed by dropping the first column instead. We are now ready to tackle the special matrix structure discussed in Section 3.1, specifically, the eigenvalues of the random matrix $H^+H' = H^\top(HH^\top)^{-1}H'$ for large N.

These constructions can be expressed as matrix products as follows:

$$H = \overline{H}W, \quad H' = \overline{H}W', \qquad \text{where} \quad W = \begin{bmatrix} I \\ 0 \end{bmatrix}, \quad W' = \begin{bmatrix} 0 \\ I \end{bmatrix}.$$

Since cyclic permutations of matrix products preserve the spectral radius, we have

$$\rho\left(H^+H'\right) = \rho\left(W'W^\top\overline{H}^\top\left(HH^\top\right)^{-1}\overline{H}\right).$$

We estimate the right side using the general matrix relations

$$\rho(M) \leq \|M\| = \sigma_{\max}(M) = \sqrt{\rho(M^\top M)}.$$

Thus we have

$$\rho\left(H^+H'\right)^2 \leq \rho\left(\overline{H}^\top\left(HH^\top\right)^{-1}\overline{H}\right). \tag{9}$$

We recognize the inverse matrix filling the sandwich on the right side from Proposition 3.3; the symmetry and nested structure in Eq. (9) supports the following result. (See Appendix B for the proof.)

Theorem 3.4. *Suppose each of the independent random variables ω_i has a standard normal distribution. Then for each fixed $L \geq 1$, there is a sequence $\epsilon_0, \epsilon_1, \ldots$, with $\epsilon_N \to 0$ as $N \to \infty$, such that*

$$\lim_{N\to\infty} \mathbb{P}\left\{\rho\left(H^+_{L,N}H'_{L,N}\right) < 1 + \epsilon_N\right\} = 1.$$

3.3. Hankel models with additive noise

We are now ready to re-introduce the system dynamics and establish the stability of the resulting Hankel-based models following Eq. (6). To provide intuition for the concept, Fig. 2 illustrates Corollary 3.5 and Theorem 3.4 by visualizing the recursion in Eq. (6).

Corollary 3.5 (Stable Hankel dynamics models)**.** *In addition to Assumptions 2.1 to 2.3, take $L \geq n$ and assume the outputs y_t have additive Gaussian noise ω_t. Assume the probing signal u is bounded for all time and satisfies:*

$$\lambda_{\min}\left(H_L(u)H_L(u)^\top\right) \to \infty.$$

Denote H to be the principal matrix in Eq. (2). Then the result of Theorem 3.4 still holds.

Remark 3.6. *A truncated Gaussian probing signal satisfies the hypotheses by Proposition 3.3.* ◂

Proof. Let z' be the last column of H':

$$z' = \left[u_{N-L} \dots u_{N-1}, y_{N-L} + \omega_{N-L} \dots y_{N-1} + \omega_{N-1}\right]^\top.$$

Using the relation $H^+ = \lim_{\delta \downarrow 0} H^\top \left(HH^\top + \delta I\right)^{-1}$ to modify Eq. (B.4) in Theorem 3.4, we arrive at the general inequality

$$\rho(H^+H') \leq \sigma_{\max}\left(H^+H'\right) \leq \sqrt{1 + \|z'\|^2 \|H^+\|^2}. \tag{10}$$

We are interested in the top $L + n$ singular values of H. Following Proposition 3.3 and Coulson et al. [27], it can be shown that $\sigma_{L+n}(H) \to \infty$, meaning $\|H^+\| \to 0$. However, this is insufficient for Eq. (10) to converge to 1. Since the system of interest is bounded-input, bounded-output stable, we have

$$\left\|z'\right\|^2 \leq 2LC + \left\|\omega'\right\|^2,$$

where C is a constant and ω' is the L-dimensional vector of noise terms in z'. The result then follows. □

Remark 3.7. *If the system is unstable, then we expect an exponential increase in the magnitude of z', blowing up our spectral radius estimate in Eq.* (10). ◁

4. Stabilizing reinforcement learning control

The YK parameterization in Eq. (4) features two ingredients for the set of stabilizing controllers: the dynamics P and the stable operator Q. The previous two sections showed how to incorporate Willems' lemma to characterize P. This leaves Q as the "learnable" component for an RL agent. The advantage of learning Q over a standard feedback policy is it enables an RL agent to update its policy in an unconstrained fashion without risking instability during training.

4.1. Learning stable operators

The Q parameter is a dynamical system. Therefore, Q is characterized by inputs, outputs, and some stable internal transition. We demonstrate two approaches for modeling stable internal dynamics amenable to deep learning and optimization frameworks: one for the linear case, then an extension to the nonlinear setting. In both cases, we make use of Lyapunov's second method: the main idea is to embed a trainable Lyapunov function inside the dynamic model.

Let us recall the definition of a *Lyapunov candidate function* $V\colon \mathbb{R}^n \to \mathbb{R}$: 1) V is continuous; 2) $V(z) > 0$ for all $z \neq 0$, and $V(0) = 0$; 3) There exists a continuous, strictly increasing function $\varphi\colon [0, \infty) \to [0, \infty)$ such that $V(z) \geq \varphi(\|z\|)$ for all $z \in \mathbb{R}^n$; 4) $V(z) \to \infty$ as $\|z\| \to \infty$.

(Linear operators) We consider stable linear operators of the form

$$Q \begin{cases} z_{t+1} = A_q z_t + B_q(e_t + \bar{y}_L) \\ u_t = C_q z_t + D_q(e_t + \bar{y}_L) \end{cases} \tag{11}$$

where $\bar{y}_L$ is the latest internal prediction, for example, from Algorithm 2. Therefore, the parameterization of Q is tied to the representation of stable matrices A_q. However, the explicit representation of stable matrices is unwieldy: $\mathcal{S}_n = \{A_q \in \mathbb{R}^{n\times n}\colon \rho(A_q) < 1\}$. Indeed, $\mathcal{S}_n$ is non-convex and neither open nor closed.

Fix an arbitrary square matrix $\widehat{M} \in \mathbb{R}^{n\times n}$ and a lower triangular matrix $L \in \mathbb{R}^{n\times n}$ with positive diagonal entries. Consider the transformation $M \leftarrow U\texttt{tanh}(D)V^\top$, based on the singular value decomposition $\widehat{M} = UDV^\top$, where $\texttt{tanh}$ is applied componentwise. Then the matrix $A_q = L^{-1}ML$ directly parameterizes the Lyapunov decrease condition $A_qL^{-1}L^{-\top}A_q^\top - L^{-1}L^{-\top} < 0$ under the quadratic function $V(z) = z^\top L^{-1}L^{-\top}z$. Therefore, we have the following result:

$$\mathcal{S}_n = \{L^{-1}UDV^\top L \in \mathbb{R}^{n\times n}\colon L > 0 \text{ lower triangular}, U \text{ and } V \text{ orthogonal}, \ D \text{ diagonal and } \|D\| < 1\}. \tag{12}$$

This is a corollary based on Gillis et al. [28].

(Nonlinear operators) For the problem of learning stable nonlinear operators, we adapt the method of Lawrence et al. [29]: the idea is to construct stable autonomous systems of the form $z_{t+1} = f_\theta(z_t)$ "by design" through the use of trainable Lyapunov functions. (θ represents a set of trainable weights.)

In the present setup, f_θ models the internal dynamics of a nonlinear Q parameter. For example, a control-affine model may be used with stable transition dynamics f_θ [30]. The interpretation of a nonlinear Q parameter is the same as the original motivation for $\mathcal{K}_{\text{stable}}$ in Eq. (4). The underlying interconnections remain the same, except now Q characterizes nonlinear controllers.

Two neural networks work in tandem to form a single model that satisfies the decrease condition central to Lyapunov's second method: a smooth neural network $\widehat{f_\theta}$, and a convex Lyapunov neural network V_θ. Set $\widehat{z}' = \widehat{f_\theta}(z)$ where z is the current "state" and $\widehat{z}'$ is the proposed next state. Two cases are possible: either $\widehat{z}'$ decreased the value of V or it did not. We can write out a correction to the dynamics in closed form by exploiting the convexity of V:

$$\begin{aligned} z_{t+1} &= f_\theta(z_t) \\ &\equiv \begin{cases} \widehat{f_\theta}(z_t), & \text{if } V(\widehat{f_\theta}(z_t)) \le \beta V(z_t) \\ \widehat{f_\theta}(z_t)\left(\frac{\beta V(z_t)}{V(\widehat{f_\theta}(z_t))}\right), & \text{otherwise} \end{cases} \\ &= \gamma \widehat{f_\theta}(z_t), \text{ where} \\ \gamma = \gamma(z_t) &= \frac{\beta V(z_t) - \texttt{ReLU}(\beta V(z_t) - V(\widehat{f_\theta}(z_t)))}{V(\widehat{f_\theta}(z_t))}. \end{aligned} \tag{13}$$

(Recall $\texttt{ReLU}(x) = \max\{0, x\}$.) Since Eq. (13) composes the model f_θ, both $\widehat{f_\theta}$ and V_θ are trained in unison towards whatever goal is required of the sequential states $z_t, z_{t+1}, \ldots$, such as supervised learning tasks. Moreover, although the model f_θ is constrained to be stable, it is unconstrained in parameter space, making its implementation and training fairly straightforward with deep learning libraries.

4.2. Unconstrained reinforcement learning over stable operators

The YK parameterization is appealing for learning-based control schemes such as RL because the closed-loop system is stable for every choice of the Q parameter. Therefore, stability does not rely on hyperparameter selection or optimality. This is in contrast to simply selecting a feedback controller without enforcing stability; see Fig. 1. Since any practical objective will require closed-loop stability, it is reasonable to allow an RL agent to manipulate the Q parameter directly. A brief overview of deep RL will serve to unify this paper, however, a thorough introduction is beyond its scope.

RL is an optimization-driven framework for learning "policies" simply through interactions with an environment [1, 2]. The states s and actions a belong to the state and action sets $\mathcal{S}$, $\mathcal{A}$, respectively. At each time step t, the state s_t influences the sampling of an action $a_t \sim \pi(\cdot \mid s_t)$ from the "policy" π. Given the action a_t, the environment produces a successor state s_{t+1}, which induces a conditional density function $s_{t+1} \sim p(\cdot \mid s_t, a_t)$ for any initial distribution $s_0 \sim p_0(\cdot)$. The desirability of a given action is quantified by a "reward" $r_t = r(s_t, a_t)$ associated with each step in the process above. This cycle produces one step in a Markov decision process. As time marches forward under a policy π, a "rollout" emerges, denoted $h = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$. Each fixed policy π induces a probability density $p^\pi(\cdot)$ on the set of rollouts.

With these pieces in place, the overall goal of the agent is to determine a policy that maximizes the cumulative discounted reward. That is, given some constant $\gamma \in (0, 1)$, the agent seeks π to

$$\begin{aligned} &\text{maximize} \quad J(\pi) = \mathbb{E}_{h \sim p^\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \\ &\text{over all} \quad \text{policies } \pi \colon \mathcal{S} \to \mathcal{P}(\mathcal{A}), \end{aligned} \tag{14}$$

where $\mathcal{P}(\mathcal{A})$ denotes the set of probability measures on $\mathcal{A}$.

In the space of all possible policies, the optimization is performed over a subset parameterized by some vector θ. In this work, the policy is the Q parameter outlined in Section 4.1. Therefore, Eq. (14) automatically satisfies an internal stability constraint over the whole weight space θ. We are then able to use any RL algorithm to solve the problem.

The broad subject of RL concerns iterative methods for choosing a desirable policy π (this is the "learning"), guided in some fundamental way by the agent's observations of the rewards from past state-action pairs (this provides the "reinforcement"). A standard approach to solving Problem (14) uses gradient ascent

$$\theta \leftarrow \theta + \eta \nabla J(\theta), \tag{15}$$

where $\eta > 0$ is a step-size parameter. Analytic expressions for $\nabla J(\theta)$ exist for both stochastic and deterministic policies [31]. However, $\nabla J(\theta)$ cannot be evaluated precisely, as it depends on the dynamics, policy, and chosen time horizon, not to mention the noise. Therefore, RL algorithms differ based on how they approximate the update scheme in Eq. (15).

Since our framework decouples stability from the learning process, one may employ any off-the-shelf RL algorithm. Therefore, as the field of deep RL matures, this stabilizing framework will remain relevant. The only requirement is an appropriate policy representation. Both the linear and nonlinear cases discussed in Section 4.1 can be implemented in a standard RL library: one must store the internal state z_t and input $e_t + \bar{y}_L$, then employ automatic differentiation to update the A_q, B_q, C_q, D_q matrices in Eq. (11) or, in the nonlinear case, f_θ in a control-affine setup. Further details are provided in Appendix A.

5. Simulation studies

We now demonstrate the proposed stabilizing framework in a series of simulation studies. We give an industrial example, showing how one can layer the stabilizing strategy on top of existing controllers. Then, we show how the ideas presented above can be adapted to directly modify fixed-structure controllers while ensuring stability. In all the examples, we use the TD3 algorithm [32]. This choice is primarily to illustrate the applicability of the framework to general algorithms. Note that the choice of RL algorithm is essentially a hyperparameter layered on top of the stable behavior it modifies. Code is available here: https://github.com/NPLawrence/StableBehavior.jl

5.1. An industrial example

The authors' industrial partner built a hardware platform to use for testing various control methods. The equipment involves a tank holding water, positioned above a second tank used as a reservoir. Water drains from the upper tank into the reservoir through an outflow pipe, while being replenished by water pumped up from the reservoir. The problem is to control the water level in the upper tank.

Two proportional-integral-derivative (PID) controllers are in operation. First, a "level controller" measures the actual water level outputs the desired inflow rate. Second, a "flow controller" uses the desired and actual inflow rates to determine the pump speed. For our purposes, both these controllers are fixed and a part of the environment.

We have reliable numerical models for all aspects of the equipment described above. The flow dynamics, based on Bernoulli's equation and conservation of fluid, are nonlinear. Low-pass filtering leads to a stream of four scalar signals: the water level, drainage flow rate, pump speed, and incoming flow rate. A full account of the apparatus and the differential equations we use to model it appears in [33]. For the results presented here, we used the simulator rather than the laboratory system. This involved discretizing the continuous dynamics cited above with time steps of 0.5 seconds and adding Gaussian measurement noise with variance 0.015.

We use the proposed stabilizing framework to generate additive corrections to the command produced by the given level PID controller. Since the environment includes a PID controller, we modify the control scheme to be in incremental form $u_t = u_{t-1} + \Delta u_t$, where Δu_t is the sum of the nonlinear YK parameter from Section 4.1 and PID controller outputs:

$$\Delta u_t = \Delta u_t^{(q)} + \Delta u_t^{(\mathrm{PID})}$$

Although the control system contains several cascaded filter terms, the full flow setpoint to measured level dynamics is approximately a first-order plus dead time system [33]. Recall Willems' lemma only requires an upper bound of the system order. We take $L = 11$ to ensure input–output trajectories are sufficiently long to capture the current dynamics in the presence of output noise. We ran 20 training sessions, each of 100 episodes. Figure 3 illustrates the cumulative rewards observed. The median over the 20 sessions provides the solid line; the interquartile ranges delimit the shaded region. We note that the median reward curve is much closer to the upper limit of the shaded region than the lower,

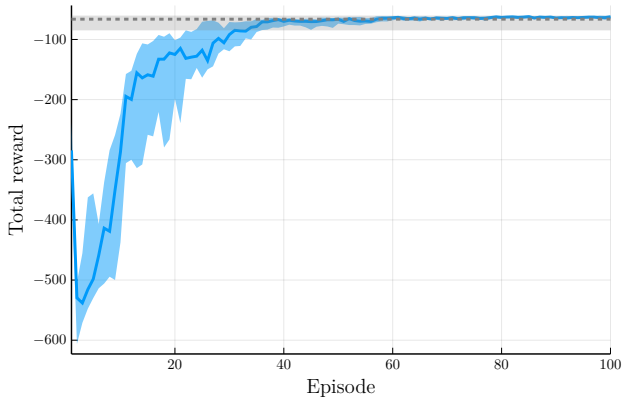

Figure 3: Cumulative reward curve over 20 training sessions. The solid line is the median and the shaded region shows the interquartile range. The dashed line and its shaded region are the final results of training without the stability constraint.

indicating that the majority of experiments fall within that tight region. Although there is significant change in the first few episodes, due to the random policy initialization, the training sessions exhibit consistent convergence. The reward curves tend to plateau after around 40 episodes. Figure 4 shows a single rollout from one of the experiments.

5.2. Direct tuning of fixed-structure controllers

In the introduction, we highlighted the potentially dangerous dependence of closed-loop stability on hyperparameter settings. So far we have decoupled stability and learning algorithms through a data-driven control scheme. However, one may wish to enforce a fixed-structure control law. We show how our framework can also deal with this case through a data-driven constraint.

Theorem 5.1 (SISO case of Furieri et al. [34])**.** *Consider the set of scalar-valued transfer functions X, Y, W satisfying the linear relation*

$$\begin{aligned} X + PY &= I \\ W - PX &= 0. \end{aligned} \tag{16}$$

Then

$$\mathcal{K}_{stable} = \left\{ YX^{-1} : \textit{Eq. (16) holds and } X, Y, W \textit{ are stable} \right\}. \tag{17}$$

Remark 5.2. *By identifying $X = \frac{1}{1+PK}, Y = \frac{K}{1+PK}, W = \frac{P}{1+PK}$, we see that Eq. (17) implicitly parameterizes all stable sensitivity functions in Section 2.2.* ◂

In contrast to the YK parameterization in Eq. (4), Theorem 5.1 characterizes the set of stabilizing controllers through the affine constraint in Eq. (16). This alternative representation is useful for imposing a desired controller structure through the variables X, Y while enforcing closed-loop stability by insisting X, Y, W be stable. We use the linear parameterization from Eq. (12).

In the behavioral setting, we propose to traverse Eq. (17) through the use of Algorithm 1 and the set of stable parameters in Eq. (11). Concretely, we generate the left-hand side of Eq. (16) by taking the outputs of X, Y, W as inputs to the Hankel-based model in Algorithm 1. We minimize the residual from the right-hand side to generate a stabilizing controller. Note this approach can be used to find an initial stabilizing controller, to be deployed in combination with the control scheme shown in Section 5.1.

(Training) Consider a plant whose continuous-time transfer function is

$$P(s) = \frac{1 - s}{(s + 1)^3}.$$

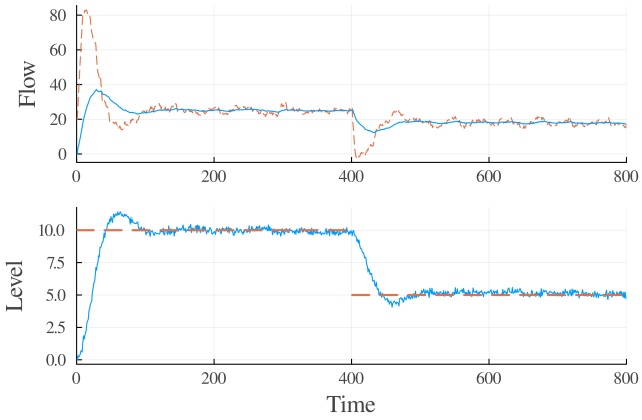

Figure 4: A sample input-output rollout by the trained RL agent for one of the training sessions. Dashed lines are setpoints; solid lines are measured values.

Like the previous example, we discretize in time and take the resulting system as the true dynamics.

We illustrate the data-driven stability constraint on a proportional-integral (PI) tuning task. The deterministic policy has the form

$$\pi_{\mathrm{PI},k_p,k_i}(s_t) = k_p(e_t - e_{t-1}) + k_i e_t \Delta t + u_{t-1},$$

where the constants k_p, k_i are parameters that we will use RL to determine. We adopt a PI controller structure for two reasons: such configurations are widely used in practice, and even this simple structure can illustrate the challenges associated with stability while achieving excellent performance in RL tasks.

We run two RL-based experiments: one with no stability constraint and one where stability is enforced by a projection-based update scheme. For the second, we project the parameter vector $\widehat{\theta} = [k_p, k_i]$ proposed by the RL algorithm by solving the optimization problem below:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \left\| \theta - \widehat{\theta} \right\| \\ \text{subject to} \quad & \pi_{\mathrm{PI},\theta} \in \mathcal{K}_{\mathrm{stable}}. \end{aligned} \tag{18}$$

Figure 1 illustrates the training performance of the two experiments. We implemented a sparse reward function by defining $r(s_t) = 1$ if $|e_t| < \delta$ and 0 otherwise, where δ is a small constant. Moreover, we ran the RL algorithm 10 times for each experiment using the default hyperparameters. We actually tweaked the actor learning rate for the unconstrained experiment to make the results more competitive. Although it is possible to improve the unconstrained results through trial and error, this underscores the importance of stability-based methods. Imposing the minimal intervention in Eq. (18) avoids the dangerous, low-reward regions altogether.

Figure 5 accompanies the constrained experiment that produced Fig. 1. It shows the distribution of PI parameters over the 10 training sessions. Specifically, we depict pre-projection (red) and post-projection (blue) values. (We removed parameter values that did not move substantially to avoid mixing the red and blue regions.) The grey curve shows the stability boundary for the underlying system. We avoided overlapping the blue region with the boundary by constraining the maximum eigenvalue of the optimization variables X, Y, W in Eq. (17). In contrast, the parameters corresponding to the unconstrained experiments in Fig. 1 (*not* the red values in Fig. 5) can leave the interior, then either recover automatically or not at all.

6. Discussion and conclusions

6.1. Extension to MIMO and unstable systems

The ingredients put forth here can, in principle, handle multiple-input, multiple-output (MIMO) and unstable systems. Willems' fundamental lemma applies for LTI systems, regardless of dimension, and makes no claims about

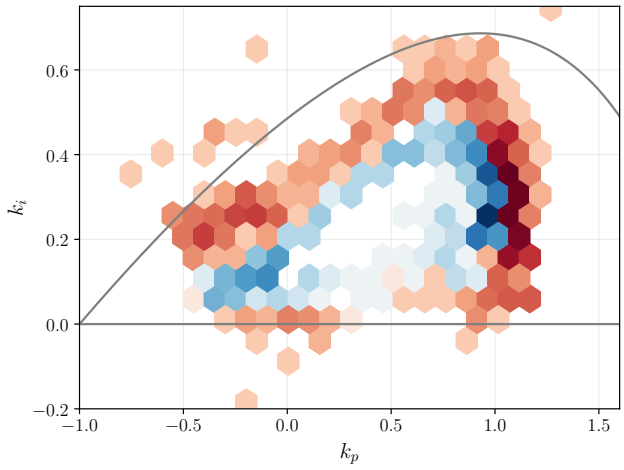

Figure 5: Heatmap of projected PI parameters strictly inside the stability boundary.

stability. The YK parameterization, as presented in Section 2.2, also applies to MIMO systems. However, the intuitive derivation given there does not apply to unstable systems. Nonetheless, the most straightforward approach is to apply the constraint-based characterization of stabilizing controllers due to Furieri et al. [34] utilized in Section 5.2. (This is in contrast to the "classical" YK approach of factorizing the plant P.) Such an approach does not require the plant to be stable, but the controller is no longer characterized in closed form using a single free parameter Q.

The constraint-based approach can be used to obtain an initial stabilizing controller. This controller can then be refined using RL, then follow the projection strategy detailed in Section 5.2. Alternatively, one may opt for a data-driven control strategy. To apply the YK parameterization in an analogous way to that in Section 2.2, one may then augment the output of the initial stabilizing controller by adding the Q parameter, such as in Section 5.1. One confounding factor in both of these approaches is collecting appropriate data from an unstable system and reliably generating rollouts from the Hankel-based model. This requires extending the results in Section 3 and is a promising avenue for future work.

6.2. Conclusion

The YK parameterization is well-known in control theory but seemingly under-utilized in RL. Taking it as a starting point, we have adapted advances in deep learning and behavioral systems to develop an end-to-end framework for learning stabilizing policies with general RL algorithms. These core ingredients invite a modular approach to learning stabilizing controllers in which past, present, and future components are cross-compatible. For example, the nonlinear Q parameterization in Section 4.1 is *functional*, rather than *structural*: as long as smoothness and the Lyapunov hypotheses are satisfied, one has freedom in terms of activations, layers, or architecture altogether. Alternatively, one may also elect to use "classical" approaches—simpler learning algorithms, restricted sets of linear operators, or observer-based control instead of employing Willems' lemma as an internal model—in combination with newer ones.

There are many further avenues to explore. These include the use of stochastic policies, extensions to unstable systems, and balancing the persistence of excitation assumption during training and steady-state operations. We believe this is a fruitful area to investigate further as deep RL gains traction in process systems engineering.

Acknowledgement

We gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and Honeywell Connected Plant. We would also like to thank Professor Yaniv Plan for helpful discussions.

References

[1] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, I. Palunko, Reinforcement learning for control: Performance, stability, and deep approximators, Annual Reviews in Control 46 (2018) 8–28.

[2] R. Nian, J. Liu, B. Huang, A review on reinforcement learning: Introduction and applications in industrial process control, Computers & Chemical Engineering 139 (2020) 106886.

[3] B. D. Anderson, From Youla–Kucera to identification, adaptive and nonlinear control, Automatica 34 (1998) 1485–1506.

[4] I. Markovsky, F. Dörfler, Behavioral systems theory in data-driven analysis, signal processing, and control, Annual Reviews in Control (2021) S1367578821000754.

[5] R. M. Kretchmar, P. M. Young, C. W. Anderson, D. C. Hittle, M. L. Anderson, C. C. Delnero, Robust reinforcement learning control with static and dynamic stability, International Journal of Robust and Nonlinear Control 11 (2001) 1469–1500.

[6] C. W. Anderson, P. M. Young, M. R. Buehner, J. N. Knight, K. A. Bush, D. C. Hittle, Robust reinforcement learning control using integral quadratic constraints for recurrent neural networks, IEEE Transactions on Neural Networks 18 (2007) 993–1002.

[7] M. Jin, J. Lavaei, Stability-certified reinforcement learning: A control-theoretic perspective, IEEE access : practical innovations, open solutions 8 (2020) 229086–229100.

[8] M. Revay, R. Wang, I. R. Manchester, Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness, 2023.

[9] F. Gu, H. Yin, L. E. Ghaoui, M. Arcak, P. Seiler, M. Jin, Recurrent neural network controllers synthesis with stability guarantees for partially observed systems, Proceedings of the AAAI Conference on Artificial Intelligence 36 (2022) 5385–5394.

[10] R. Wang, N. H. Barbara, M. Revay, I. R. Manchester, Learning over all stabilizing nonlinear controllers for a partially-observed linear system, IEEE Control Systems Letters 7 (2022) 91–96.

[11] H. Zhang, L. Cui, X. Zhang, Y. Luo, Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method, IEEE Transactions on Neural Networks 22 (2011) 2226–2236.

[12] H. Modares, F. L. Lewis, M.-B. Naghibi-Sistani, Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems, Automatica 50 (2014) 193–202.

[13] F. Berkenkamp, M. Turchetta, A. Schoellig, A. Krause, Safe model-based reinforcement learning with stability guarantees, in: Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017, pp. 1–11.

[14] M. Han, L. Zhang, J. Wang, W. Pan, Actor-critic reinforcement learning for control with stability guarantee, IEEE Robotics and Automation Letters 5 (2020) 6217–6224.

[15] Y. Kim, J. M. Lee, Model-based reinforcement learning for nonlinear optimal control with practical asymptotic stability guarantees, AIChE Journal 66 (2020).

[16] Y.-C. Chang, S. Gao, Stabilizing neural control using self-learned almost Lyapunov critics, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 1803–1809.

[17] S. Gros, M. Zanon, Learning for MPC with stability & safety guarantees, Automatica 146 (2022) 110598.

[18] J. Perdomo, J. Umenberger, M. Simchowitz, Stabilizing dynamical systems via policy gradient methods, in: Advances in Neural Information Processing Systems, volume 34, Curran Associates, Inc., 2021, pp. 29274–29286.

[19] S. Lale, K. Azizzadenesheli, B. Hassibi, A. Anandkumar, Reinforcement learning with fast stabilization in linear dynamical systems, in: Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, PMLR, 2022, pp. 5354–5390.

[20] S. Mukherjee, T. L. Vu, Reinforcement learning of structured stabilizing control for linear systems with unknown state matrix, IEEE Transactions on Automatic Control (2022) 1–1.

[21] J. W. Roberts, I. R. Manchester, R. Tedrake, Feedback controller parameterizations for reinforcement learning, in: 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), IEEE, Paris, 2011, pp. 310–317.

[22] S. R. Friedrich, M. Buss, A robust stability approach to robot reinforcement learning based on a parameterization of stabilizing controllers, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Singapore, Singapore, 2017, pp. 3365–3372.

[23] J. C. Willems, P. Rapisarda, I. Markovsky, B. L. De Moor, A note on persistency of excitation, Systems & Control Letters 54 (2005) 325–329.

[24] H. J. van Waarde, C. De Persis, M. K. Camlibel, P. Tesi, Willems' fundamental lemma for state-space systems and its extension to multiple datasets, IEEE Control Systems Letters 4 (2020) 602–607.

[25] J. Berberich, F. Allgower, A trajectory-based framework for data-driven system analysis and control, in: 2020 European Control Conference (ECC), IEEE, Saint Petersburg, Russia, 2020, pp. 1365–1370.

[26] M. Rudelson, R. Vershynin, Hanson-Wright inequality and sub-Gaussian concentration, Electronic Communications in Probability 18 (2013).

[27] J. Coulson, H. J. Van Waarde, J. Lygeros, F. Dörfler, A quantitative notion of persistency of excitation and the robust fundamental lemma, IEEE Control Systems Letters 7 (2022) 1243–1248.

[28] N. Gillis, M. Karow, P. Sharma, Approximating the nearest stable discrete-time system, Linear Algebra and its Applications 573 (2019) 37–53.

[29] N. P. Lawrence, P. D. Loewen, M. G. Forbes, J. U. Backström, R. B. Gopaluni, Almost surely stable deep dynamics, in: Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 18942–18953.

[30] E. D. Sontag, et al., Smooth stabilization implies coprime factorization, IEEE transactions on automatic control 34 (1989) 435–443.

[31] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, volume 32, PMLR, PMLR, 2014, pp. 387–395.

[32] S. Fujimoto, H. van Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: Proceedings of the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, PMLR, 2018, pp. 1587–1596.

[33] N. P. Lawrence, M. G. Forbes, P. D. Loewen, D. G. McClement, J. U. Backström, R. B. Gopaluni, Deep reinforcement learning with shallow controllers: An experimental application to PID tuning, Control Engineering Practice 121 (2022) 105046.

[34] L. Furieri, Y. Zheng, A. Papachristodoulou, M. Kamgarpour, An Input-Output Parametrization of Stabilizing Controllers: Amidst Youla and System Level Synthesis, IEEE Control Systems Letters 3 (2019) 1014–1019.

[35] J. Tian, o. contributors, ReinforcementLearning.jl: A reinforcement learning package for the Julia programming language, 2020.

[36] F. B. Carlson, M. Fält, A. Heimerson, O. Troeng, ControlSystems. jl: A control toolbox in Julia, in: 2021 60th IEEE Conference on Decision and Control (CDC), IEEE, 2021, pp. 4847–4853.
[37] B. Legat, O. Dowson, J. D. Garcia, M. Lubin, MathOptInterface: A data structure for mathematical optimization problems, INFORMS Journal on Computing 34 (2022) 672–689.

Appendix A. Implementation details

Numerical experiments were carried out in the Julia programming language. We utilized `ReinforcementLearning.jl` [35], `ControlSystems.jl` [36], and `NLOpt.jl` [37].

As discussed in Section 4, any RL algorithm may be employed as long as the user provides an appropriate Q parameterization to represent the policy. For approaches based on random search or direct methods, one may simply generate rollouts via Algorithm 2 inside an optimization program. However, this strong dependence between rollouts and policy parameters can break when using policy gradient-based methods. We take the LTI case in Eq. (11) as an example. If one stores z_t and $e_t+\bar{y}_L$ as the RL state, then training the policy—that is, the Q parameter—as $\pi(s_t) = C_q z_t + D_q(e_t+\bar{y}_L)$ will not result in updates to the A_q and B_q matrices. Therefore, even though the environment can be rolled out with Eq. (11), the policy requires z_t to *explicitly* be a function of A_q and B_q, namely, by unrolling Eq. (11) for one time step. Once the policy is written in an appropriate fashion, policy gradient-based RL implementations can automatically compute each gradient component $\frac{\partial \pi}{\partial \theta_i}$ with θ being a vector of all components in A_q, B_q, C_q, D_q.

Appendix B. Further details on random Hankel matrices

Proposition 3.3. *Suppose each of the independent random variables ω_i has a standard normal distribution. Then for each fixed $L \geq 1$, there is a sequence $r_0, r_1, \ldots$, with $r_N \to \infty$ as $N \to \infty$, such that*

$$\lim_{N\to\infty} \mathbb{P}\left\{\sigma_{\min}\left(H_{L,N}\right) > r_N\right\} = 1.$$

Here $\sigma_{\min}(\cdot)$ returns the smallest singular value of its matrix argument.

Proof. First consider some realization of $\mathcal{H}$ and specific values of L, N, with $N \geq L$. Simplify notation by writing $H = H_{L,N}$ for the specific $L \times N$ Hankel matrix of interest, and let its rows define the N-component vectors

$$\overline{\omega}_i = \begin{bmatrix} \omega_i & \omega_{i+1} & \cdots & \omega_{i+N-1} \end{bmatrix}, \qquad i = 0, 2, \ldots, L-1. \tag{B.1}$$

Then $\sigma_{\min}(H)^2$ is the smallest eigenvalue of the $L \times L$ matrix $HH^\top$.

To estimate this minimum eigenvalue, we split the matrix of interest as $HH^\top = D + R$, where D is the diagonal and $R = HH^\top - D$ is the remainder. (It is helpful to write down these matrices in terms of Eq. (B.1) for reference.) $HH^\top$ is symmetric, so the variational characterization of eigenvalues gives

$$\lambda_{\min}(HH^\top) = \lambda_{\min}(D+R) \geq \lambda_{\min}(D) + \lambda_{\min}(R).$$

We expect that $\lambda_{\min}(D)$ is "large", and $|\lambda_{\min}(R)|$ is "small". Let us quantify these intuitions under two preliminary conditions. Assume first that some fixed real parameter θ dominates the magnitude of every entry in R, that is,

$$\left|\left\langle \overline{\omega}_i,\ \overline{\omega}_j \right\rangle\right| \leq \theta N, \qquad \forall i, j \in [0, L-1] \text{ with } i \neq j. \tag{B.2}$$

Next, assume that some $\alpha \in (0,1)$ obeys

$$\omega_{L-1}^2 + \ldots + \omega_{N-1}^2 \geq \alpha(N - L + 1). \tag{B.3}$$

For the matrix R, Gershgorin's Circle Theorem implies

$$\lambda_{\min}(R) \geq \min_i \left(- \sum_{j \neq i} \left|\left\langle \overline{\omega}_i,\ \overline{\omega}_j \right\rangle\right| \right) \geq -N\theta(L-1).$$

For the matrix D, each diagonal entry is a sum of N squares. Every such sum includes the $N - L + 1$ terms on the left side of Eq. (B.3). Thus Eq. (B.3) provides a lower bound for every diagonal entry in D, and of course one of those diagonal entries is the smallest. We deduce that

$$\lambda_{\min}(D) \geq \omega_{L-1}^2 + \ldots + \omega_{N-1}^2 \geq \alpha(N - L + 1).$$

We conclude that

$$\lambda_{\min}(HH^\top) \geq \alpha(N - L + 1) - N\theta(L - 1).$$

With the specific choices

$$\theta = \frac{1}{L+1}, \qquad \alpha = \frac{L}{L+1},$$

we have both $\alpha, \theta \in (0, 1)$ and $\alpha = L\theta$, leading to

$$\lambda_{\min}(HH^\top) \geq N\theta - \theta L(L-1) = \frac{N}{L+1} - \frac{L(L-1)}{L+1}.$$

Define $r_N > 0$ by matching r_N^2 with the right side here. Then $\sigma_{\min}(H) \geq r_N$.

Continuing with fixed N and L, let us now estimate the probabilities of the prerequisite inequalities above. In condition Eq. (B.2), the inner product fits the pattern in Corollary 3.2(b), and we have

$$\mathbb{P}\left\{\left|\left\langle \overline{\omega}_i,\ \overline{\omega}_j \right\rangle\right| \leq N\theta\right\} \geq 1 - 2\exp\left(-c_1\theta^2 N\right).$$

For condition Eq. (B.3), Corollary 3.2(a) gives

$$\begin{aligned}\mathbb{P}\left\{\omega_{L-1}^2 + \ldots + \omega_{N-1}^2 \geq \alpha(N - L + 1)\right\} \\ \geq 1 - 2\exp\left(-c_0\alpha^2(N - L + 1)\right).\end{aligned}$$

As $N \to \infty$, the $L(L-1)/2$ events in Eq. (B.2) and the further condition in Eq. (B.3) have probabilities that converge to 1 exponentially quickly. The same must be true of their intersection, and we have shown that this covers the situation where $\sigma_{\min}(H) \geq r_N$. This completes the proof. □

Theorem 3.4. *Suppose each of the independent random variables ω_i has a standard normal distribution. Then for each fixed $L \geq 1$, there is a sequence $\epsilon_0, \epsilon_1, \ldots$, with $\epsilon_N \to 0$ as $N \to \infty$, such that*

$$\lim_{N\to\infty} \mathbb{P}\left\{\rho\left(H_{L,N}^+ H'_{L,N}\right) < 1 + \epsilon_N\right\} = 1.$$

Proof. Let us write $\omega' = (\omega_{N-L}, \ldots, \omega_{N-1})$ for the last column in $\overline{H}$, to create the block-structured expression $\overline{H} = [H\ \omega']$. Then, using cyclic permutation and the upper bound in Eq. (9),

$$\begin{aligned}\rho\left(H^+H'\right)^2 &\leq \rho\left(\overline{H}^\top \left(HH^\top\right)^{-1} \overline{H}\right) \\ &= \rho\left(\left(HH^\top\right)^{-1} \overline{H}\,\overline{H}^\top\right) \\ &= 1 + \rho\left(\left(HH^\top\right)^{-1} \omega'\omega'^\top\right).\end{aligned}$$

Here the final equation holds because the matrix added to I in the line above is positive semi-definite, being the product of two factors that are each positive semi-definite and symmetric. Indeed, these same two properties support the following estimate:

$$\begin{aligned}\rho\left(\left(HH^\top\right)^{-1}\omega'\omega'^\top\right) &\le \left\|\left(HH^\top\right)^{-1}\right\| \left\|\omega'\omega'^\top\right\| \\ &= \frac{\rho(\omega'\omega'^\top)}{\min_{x\neq 0}\frac{\|HH^\top x\|}{\|x\|}} \\ &= \frac{\omega'^\top\omega'}{\lambda_{\min}\left(HH^\top\right)}.\end{aligned}$$

We arrive at the intermediate result

$$\rho(H^+H') \le \sigma_{\max}\left(H^+H'\right) \le \sqrt{1+\frac{\|\omega'\|^2}{\lambda_{\min}\left(HH^\top\right)}}. \tag{B.4}$$

Here Proposition 3.3 is relevant. Let $\{r_N\}$ be a sequence with $r_N \to \infty$ for which

$$\mathbb{P}\left\{\lambda_{\min}(HH^\top) \ge r_N\right\} \to 1. \tag{B.5}$$

Invent any sequence δ_N with $\delta_N \to 0$ such that $r_N\delta_N \to \infty$. Manipulate random events as follows:

$$\begin{aligned}\left\{\frac{\|\omega'\|^2}{\lambda_{\min}(HH^\top)} > \delta_N\right\} &= \left\{\frac{\|\omega'\|^2}{\lambda_{\min}(HH^\top)} > \delta_N\right\} \cap \left(\left\{\lambda_{\min}(HH^\top) \le r_N\right\} \cup \left\{\lambda_{\min}(HH^\top) > r_N\right\}\right) \\ &\subseteq \left\{\lambda_{\min}(HH^\top) \le r_N\right\} \cup \left\{\frac{\|\omega'\|^2}{r_N} > \delta_N\right\}.\end{aligned}$$

The first event on the right is controlled by Eq. (B.5), while Markov's inequality gives

$$\mathbb{P}\left\{\frac{\|\omega'\|^2}{r_N} > \delta_N\right\} \le \frac{\mathbb{E}\|\omega'\|^2}{\delta_N r_N} = \frac{L^2}{\delta_N r_N} \to 0.$$

In view of Eq. (B.4), we have

$$\mathbb{P}\left\{\sigma_{\max}\left(H^+H'\right) > \sqrt{1+\delta_N}\right\} \to 0 \qquad \text{as } N \to \infty.$$

The stated result is an elementary reformulation of this. □