Hyperparameter Tuning of an Off-Policy Reinforcement Learning Algorithm for H_∞ Tracking Control

Alireza Farahmandi ALIREZA.FARAHMANDI.CIV@US.NAVY.MIL
Brian Reitz BRIAN.C.REITZ2.CIV@US.NAVY.MIL
Mark Debord MARK.J.DEBORD.CIV@US.NAVY.MIL
Douglas Philbrick DOUGLAS.O.PHILBRICK2.CIV@US.NAVY.MIL
Katia Estabridis KATIA.N.ESTABRIDIS.CIV@US.NAVY.MIL
Gary Hewer GARY.A.HEWER.CIV@US.NAVY.MIL
Naval Air Warfare Center Weapons Division, China Lake, CA 93555, USA

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

In this work, we present the hyperparameter optimization of an online, off-policy reinforcement learning algorithm based on a parallel search. Since this model-free learning algorithm solves the H_∞ optimal tracking problem iteratively using ordinary least squares regression, we propose using the condition number of the data matrix as a model-free measure for tuning the hyperparameters. This addition enables automated optimization of the involved hyperparameters. We demonstrate that the condition number is a useful metric for tuning the number of collected samples, sampling interval, and other hyperparameters involved. In addition, we demonstrate a correlation between this condition number and properties of the sum of sinusoids persistent excitation.

Keywords: Reinforcement learning, Hyperparameter tuning, Condition number, Optimization, Infinite horizon optimal control

1. Introduction

In machine learning, hyperparameters are parameters used to control the learning procedure. They configure various aspects of the learning algorithm and can have widely varying effects on the results and performance. Hyperparameters must be set appropriately by the user in order to maximize the usefulness of the learning approach (Claesen and Moor, 2015). Hyperparameter optimization, or tuning, is the process of choosing a set of hyperparameters that result in good learning performance and convergence of the learning algorithm on a particular problem. Hyperparameter search is commonly performed manually and is typically computationally expensive (Farahmandi et al., 2018). Most approaches to hyperparameter tuning involve either parallel search, sequential optimization, or some combination of the two (Cauwet et al., 2020; Bergstra et al., 2013). Parallel methods, such as grid search and random search, perform many parallel optimizations, each with a different set of hyperparameters, with a view to finding a single best output from one of the optimizations. Parallel methods exploit computational horsepower by coordination of simultaneous learning processes. Sequential optimization methods perform few optimization processes in parallel, but do so many times sequentially. Information from previous steps is used to focus the search on the most promising parameter values (e.g., Bayesian optimization) (Jaderberg et al., 2017; Turner et al., 2021).

We present for the first time (to our knowledge), the hyperparameter optimization of an online, off-policy integral reinforcement learning (IRL) algorithm based on a parallel search using condition number of the data matrix as a performance measure. We first review the construct of the H_∞ tracking problem. We discuss the iterative algorithm used to learn the control policy based solely on collected data. Next, our proposed model-free approach for tuning the number of data samples N and the sampling interval T is discussed. In order to be able to quantify differences between analytic and experimental results, we performed studies with a linear time-invariant (LTI) system, a pitch-plane model of an F-16 aircraft, followed by tuning of the persistent excitation. Finally, we show results for condition number, performance function difference between optimal and learned, and policy difference between optimal and learned for a wide range of hyperparameter values to demonstrate the dependence of the learned solution on each parameter.

2. H_∞ Tracking Problem

We begin by briefly describing the H_∞ tracking problem presented and solved in Modares et al., 2015. The reader is encouraged to review the reference for a more detailed description. Consider an LTI system[1] with dynamics represented by (1) where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the control input, $d \in \mathbb{R}^q$ is the external disturbance, and $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $D \in \mathbb{R}^{n\times q}$ are the plant, input, and disturbance matrices, respectively. It is assumed that the coefficients in A, B, and D are unknown and that the system is stabilizable.

$$\dot{x} = Ax + Bu + Dd \tag{1}$$

The tracking error is defined in (2) where $r(t)$ is the reference trajectory.

$$e_d \triangleq x(t) - r(t) \tag{2}$$

The performance output is a function of the tracking error and the control as

$$\|z(t)\|^2 = {e_d}^T Q e_d + u^T R u \tag{3}$$

where $Q \succeq 0$ and $R \succ 0$ are user-defined matrices. For the tracking problem, we define the augmented system state as

$$\bar{X} = \begin{bmatrix} {e_d}^T & r^T \end{bmatrix}^T \in \mathbb{R}^{2n}. \tag{4}$$

The goal is to attenuate the effect of the disturbance input d on the performance output z. The disturbance attenuation condition is

$$\frac{\int_t^\infty e^{-\alpha(\tau-t)}(\bar{X}^T\bar{Q}\bar{X} + u^T R u)\, d\tau}{\int_t^\infty e^{-\alpha(\tau-t)}(d^T d)\, d\tau} \le \gamma^2 \tag{5}$$

where $\alpha > 0$ is a discount factor, γ is a parameter that quantifies the amount of attenuation from the disturbance input $d(t)$ to the defined performance output variable $z(t)$, and

$$\bar{Q} = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}. \tag{6}$$

1. In this paper, we focus exclusively on LTI systems, but the original work of Modares et al., 2015 is applicable to any continuous-time system.

This disturbance attenuation condition leads to the discounted performance function

$$J(u,d) = \int_t^\infty e^{-\alpha(\tau-t)}(\bar{X}^T\bar{Q}\bar{X} + u^T R u - \gamma^2 d^T d)\,d\tau. \tag{7}$$

The objective of the H_∞ tracking problem is to minimize the discounted performance function. Considering the augmented state vector defined in (4), the system dynamics can be rewritten as

$$\dot{\bar{X}} = \bar{A}\bar{X} + \bar{B}u + \bar{D}d \tag{8}$$

with

$$\bar{A} = \begin{bmatrix} A & A - A' \\ 0 & A' \end{bmatrix}, \bar{B} = \begin{bmatrix} B^T & 0^T \end{bmatrix}^T, \bar{D} = \begin{bmatrix} D^T & 0^T \end{bmatrix}^T \tag{9}$$

where $\dot{r}(t) = A'r(t)$ is the linearized model of the command generator dynamics. The solution of this H_∞ tracking problem is determined by the following game algebraic Riccati equation (GARE):

$$\bar{Q} + (\bar{A} - 0.5\alpha I)^T P + P(\bar{A} - 0.5\alpha I) - P\bar{B}R^{-1}\bar{B}^T P + \frac{1}{\gamma^2} P\bar{D}\bar{D}^T P = 0. \tag{10}$$

The optimal control is obtained by the unique solution of the GARE (10):

$$u^* = -R^{-1}\bar{B}^T P\bar{X}. \tag{11}$$

3. Off-Policy Integral Reinforcement Learning Algorithm

Our proposed hyperparameter tuning is for the off-policy IRL algorithm presented in Modares et al., 2015. In this section, we describe this algorithm for LTI systems. It consists of two phases to solve the H_∞ optimal tracking problem for systems with unknown dynamics. These are known as the *data-collection phase* and the *learning phase*. The augmented system dynamics in (8) can be written as

$$\dot{\bar{X}} = \bar{A}\bar{X} + \bar{B}u_i + \bar{D}d_i + \bar{B}(u - u_i) + \bar{D}(d - d_i) \tag{12}$$

where u_i and d_i are the actor and disturber policies, respectively, to be updated and the i subscript denotes the current iteration of the learning algorithm. During the *data-collection phase* of the IRL algorithm, we apply a fixed stable control policy u and disturbance d to the system and collect information about the state, control input, and disturbance at N samples, taken at sampling interval T. In the *learning phase* of the algorithm, we use the collected information to iteratively solve the following off-policy IRL Bellman equation for the value function V_i, updated control policy u_{i+1}, and disturbance d_{i+1} until a stopping criterion is met:

$$\begin{aligned} e^{-\alpha T} V_i(\bar{X}(t+T)) - V_i(\bar{X}(t)) &= \int_t^{t+T} e^{-\alpha(\tau-t)}(-\bar{X}^T\bar{Q}\bar{X} - u_i^T R u_i + \gamma^2 d_i^T d_i)\,d\tau \\ &+ \int_t^{t+T} e^{-\alpha(\tau-t)}(-2u_{i+1}^T R(u - u_i) + 2\gamma^2 d_{i+1}^T(d - d_i))\,d\tau. \end{aligned} \tag{13}$$

Note that α and γ are correlated in the GARE (10) and since they are used to control the overall learning process, they are considered hyperparameters of the algorithm. According to Theorem 4 in

Modares et al., 2015, the control solution (11) makes the system (8) with $d = 0$ locally asymptotically stable if

$$\alpha \leq \alpha^* = 2\|(\bar{L}Q)^{\frac{1}{2}}\| \tag{14}$$

where

$$\bar{L} = BR^{-1}B^T + \frac{1}{\gamma^2}DD^T. \tag{15}$$

As a practical matter, a very small α and/or a large Q will satisfy condition (14). The GARE (10) effectively imposes a lower bound on γ in order to be a solution. Also important is the fact that in (13), u is the control applied to the system and can be different than the updated policy u_i.

For LTI systems with full-state feedback, knowing the form of the optimal performance function, control, and disturbance, allows us to approximate the value function, updated control policy, and updated disturbance in the Bellman equation (13) as functions of the states such that

$$V_i(\bar{X}) \approxeq \hat{W}_{v_i}^T(\bar{X} \otimes \bar{X}) \tag{16}$$

$$u_{i+1}(\bar{X}) \approxeq \hat{W}_{u_{i+1}}^T \bar{X} \tag{17}$$

$$d_{i+1}(\bar{X}) \approxeq \hat{W}_{d_{i+1}}^T \bar{X} \tag{18}$$

where $\hat{W}_{v_i} \in \mathbb{R}^{4n^2}$, $\hat{W}_{u_{i+1}} \in \mathbb{R}^{2n \times m}$ and $\hat{W}_{d_{i+1}} \in \mathbb{R}^{2n \times q}$ are time-constant weight vectors and $\otimes$ denotes the Kronecker product. Using (16), the left hand side of (13) becomes

$$\hat{W}_{v_i}^T(e^{-\alpha T}(\bar{X}(t+T) \otimes \bar{X}(t+T)) - (\bar{X}(t) \otimes \bar{X}(t))). \tag{19}$$

Using (17) and (18) for iteration i, the first term on the right hand side of (13) becomes

$$\text{vec}(-\bar{Q} - \hat{W}_{u_i}R\hat{W}_{u_i}^T + \gamma^2 \hat{W}_{d_i}\hat{W}_{d_i}^T)^T \times \int_t^{t+T} e^{-\alpha(\tau - t)}(\bar{X}(t) \otimes \bar{X}(t))\, d\tau \tag{20}$$

and the second term on the right hand side of (13) becomes

$$\begin{aligned}
&- \text{vec}(\hat{W}_{u_{i+1}})^T 2(R \otimes I(2n)) \\
&\left(\int_t^{t+T} e^{-\alpha(\tau - t)}(u \otimes \bar{X}(t))\, d\tau - (\hat{W}_{u_i}^T \otimes I(2n)) \int_t^{t+T} e^{-\alpha(\tau - t)}(\bar{X}(t) \otimes \bar{X}(t))\, d\tau \right) \\
&+ \text{vec}(\hat{W}_{d_{i+1}})^T 2\gamma^2 \left(\int_t^{t+T} e^{-\alpha(\tau - t)}(d \otimes \bar{X}(t))\, d\tau \right. \\
&\qquad \left. - (\hat{W}_{d_i}^T \otimes I(2n)) \int_t^{t+T} e^{-\alpha(\tau - t)}(\bar{X}(t) \otimes \bar{X}(t))\, d\tau \right).
\end{aligned} \tag{21}$$

Combining (19), (20), and (21) allows us to write the IRL Bellman equation (13) in the form

$$y(t) = \hat{W}^T h(t) + e(t) \tag{22}$$

where

$$y(t) = \text{vec}(-\bar{Q} - \hat{W}_{u_i}R\hat{W}_{u_i}^T + \gamma^2 \hat{W}_{d_i}\hat{W}_{d_i}^T)^T \times \int_t^{t+T} e^{-\alpha(\tau - t)}(\bar{X}(t) \otimes \bar{X}(t))\, d\tau, \tag{23}$$

$$\hat{W} = \begin{bmatrix} \hat{W}_{v_i}^T & \text{vec}(\hat{W}_{u_{i+1}})^T & \text{vec}(\hat{W}_{d_{i+1}})^T \end{bmatrix}^T, \tag{24}$$

$$h(t) = \begin{bmatrix} e^{-\alpha T}(\bar{X}(t+T) \otimes \bar{X}(t+T)) - (\bar{X}(t) \otimes \bar{X}(t)) \\ \\ 2(R \otimes I(2n)) \Big(\int_t^{t+T} e^{-\alpha(\tau - t)} (u \otimes \bar{X}(t))\, d\tau \\ -(\hat{W}_{u_i}^T \otimes I(2n)) \int_t^{t+T} e^{-\alpha(\tau - t)} (\bar{X}(t) \otimes \bar{X}(t))\, d\tau \Big) \\ \\ -2\gamma^2 \Big(\int_t^{t+T} e^{-\alpha(\tau - t)} (d \otimes \bar{X}(t))\, d\tau \\ -(\hat{W}_{d_i}^T \otimes I(2n)) \int_t^{t+T} e^{-\alpha(\tau - t)} (\bar{X}(t) \otimes \bar{X}(t))\, d\tau \Big) \end{bmatrix} \tag{25}$$

and the Bellman error $e(t)$ is the error associated with the approximations in (16)-(18).

Following the algorithm, we collect N samples of state, control input, and disturbance data at sampling interval T (points t_1 to t_N), and use that information to form

$$H = \begin{bmatrix} h(t_1), & \ldots, & h(t_N) \end{bmatrix}^T \tag{26}$$

$$Y = \begin{bmatrix} y(t_1), & \ldots, & y(t_N) \end{bmatrix}^T. \tag{27}$$

Thus, we can write

$$Y = H\hat{W} \tag{28}$$

where Y is the *observation vector* and H is the *data matrix*.

We can solve (28) for $\hat{W}$ by the Ordinary Least Squares method, and if H has full rank the solution is

$$\hat{W} = H^+ Y = (H^T H)^{-1} H^T Y \tag{29}$$

where H^+ is the pseudoinverse of the H matrix. Plugging the weight vector $\hat{W}$ into (16)-(18) allows us to compute the value function V_i, updated control policy u_{i+1}, and the updated disturbance d_{i+1}.

4. Proposed Hyperparameter Tuning

This model-free IRL algorithm solves the H_∞ optimal tracking problem without requiring any knowledge of the system dynamics. We do, however, assume that we at least know the structure of the system dynamics, so the dimensions n, m, and q are known. We also assume access to state variables and the disturbance. In this model-free sense, one might ask how to form the data matrix $H \in \mathbb{R}^{N \times M}$ where N is the number of samples with sampling interval T, and $M = 4n^2 + (m \times 2n) + (q \times 2n)$ is the number of independent coefficients in $\hat{W}$. Since M is a fixed number based on the system dynamics (16)-(18), N is the only unknown parameter that affects the size of the H matrix. In the least squares problem (28), if $N = M$ and H is nonsingular, the answer is simply $\hat{W} = H^{-1}Y$. If, however, $N > M$ so that we have more equations than unknowns, the problem is called *overdetermined* and in general no $\hat{W}$ will satisfy (28) (Demmel, 1997, §3.1). Note that, if H has full rank, the moment matrix $H^T H$ in (29) can be inverted which gives the solution.

For the numerical analysis of this least squares problem, we use *Singular Value Decomposition (SVD)* as a powerful tool for determining the quality of the data matrix. In SVD, for any $H \in \mathbb{R}^{N \times M}$, according to (Demmel, 1997, Theorem 3.2), there exist orthogonal matrices $U \in \mathbb{R}^{M \times M}$

and $V \in \mathbb{R}^{N\times N}$ such that $U^T HV = \mathrm{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{N\times M}$ where $p = \min(N, M)$ and the *singular values* of H are $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

The condition number $\kappa(H)$ is the ratio of the largest to smallest singular values ($\kappa(H) = \sigma_1/\sigma_p$). Our experiments show that the condition number of the data matrix can become quite large making the least squares problem (28) ill-conditioned. This demonstrates the sensitivity of the least squares solution to small changes in the data matrix coefficients (Golub and Van Loan, 1996, §2.7; Higham, 2002, §20.1). Since we solve the IRL Bellman equation (13) iteratively, finding the accurate solution of the least squares problem for an ill-conditioned data matrix is numerically and computationally challenging.

In order to collect data in the *data-collection phase* of the algorithm and form the data matrix (26), one needs to select N and T without requiring any knowledge of the system dynamics. We found that different selections of N and T, with all other parameters including the initial state vector x_0 fixed, yield different $\kappa(H)$ and show different properties of convergence to the optimal solution. Smaller values of $\kappa(H(N, T))$ result in better convergence to the optimal solution. Since these two parameters are used to control the overall learning process, they are also considered to be hyperparameters of the algorithm. To find the best selection of N and T, we used the *Grid Search* process that searches through a manually specified subset of the N-T hyperparameter space. Note that in this search, the initial state x_0 and all other parameters are fixed.

Through experimentation, we determined that the data matrix always has the highest condition number during the second iteration of the *learning phase* of the algorithm, so our *Grid Search* results only plot that value. As soon as the *Grid Search* is completed and we have estimates of the best values for N and T, we can then return and complete the *learning phase* using those values to solve the optimization problem. We present results for an example in the following section.

5. Model-free Numerical Tuning of N and T

In this section, we present model-free numerical tuning of the hyperparameters N and T using fixed values of α and γ for a LTI system. In Section 7, we show how our model-free tuning correlates with the optimal solution for different values of α and γ. This LTI example is taken from Modares et al., 2015 and represents the longitudinal dynamics of an F-16 in cruise coupled with an elevator actuator. The model is defined in the form of (1) with

$$A = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \quad D = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad x = \begin{bmatrix} a \\ q \\ \delta \end{bmatrix} \tag{30}$$

where a is the angle of attack, q is the pitch rate and δ is the elevator deflection angle. The objective is to design an angle-of-attack controller to track a constant reference signal such that $r = \text{const}$ and $\dot{r} = 0$. This makes $A' = 0$ in (9). Also, we have

$$Q = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad R = 1. \tag{31}$$

Now, we can form the augmented system dynamics (8) using (9). Using the Riccati solution of the GARE (10) in (11), the Euclidean norm of the optimal gain $K^* = R^{-1}\bar{B}^T P$ can be computed for a range of α and γ, as shown in Figure 1.

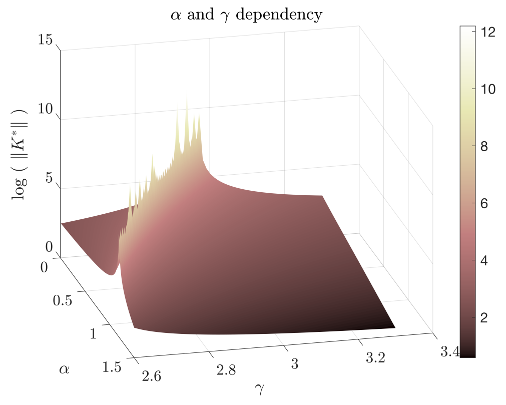
For this LTI system, from trial and error, we find that there is no solution to the Riccati equation for γ less than the lower bound $\gamma^* = 2.6$. In a model-free situation, however, one has to be able to safely pick α and γ without reference to such information. In Figure 1, we note that for $\gamma > 3$, regardless of α, the resulting $\|K^*\|$ is relatively small and decreases with an increase in γ. In Section 7, we show how results vary depending on the selection of α and γ.

Now, with a fixed set of values for α and γ (0.1 and 10, respectively), we implement the algorithm and solve the H_∞ tracking problem according to Modares et al., 2015. We can compute the upper bound $\alpha^* = 0.8944$ from (14). Data was collected using a disturbance on angle of attack modeled as a sum of 1000 sinusoids with angular frequencies from a uniform random distribution between -50 and 50 rad/s:

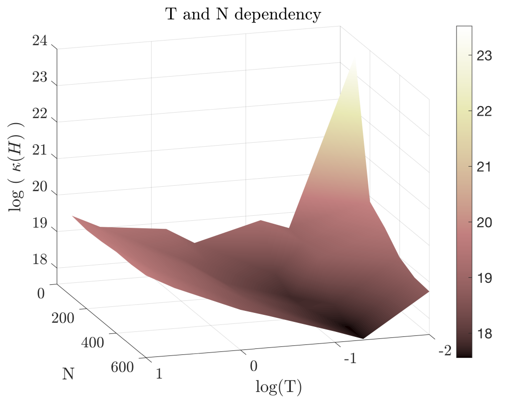
$$d(t) = 0.08 \sum_{i=1}^{1000} \sin(\omega_i t), \qquad \omega_i \in [-50, 50]. \tag{32}$$

Also, because the control input and all of the states need to be observable in the data matrix, we persistently excite the control input and the command generator states with a sum of sinusoids in the same format as (32), but with appropriately selected amplitudes, while collecting the required data. These amplitudes constitute an additional set of hyperparameters that need to be tuned in order to achieve an acceptable solution.

In order to determine the best estimate of N and T, we performed a *Grid Search* for $N \in [100, 200, 300, 400, 500, 600]$ and $T \in [0.01, 0.05, 0.1, 0.5, 1, 5, 10]$. The results of this search are shown in Figure 1. The algorithm converges well to the optimal solution with the lowest condition number $\log_{10}(\kappa(H(500, 0.05))) = 17.56$ corresponding to $N = 500$ and $T = 0.05$ seconds. If we only want to collect $N = 200$ samples, the collection needs to be done at a sampling interval of $T = 0.1$ seconds, corresponding to the lowest condition number of H in the plot for $N = 200$. Also, if we want to collect data at a sampling interval of $T = 0.05$ seconds, we should collect more than $N = 300$ samples. Therefore, we are able to choose appropriate values for N and T without any knowledge of the system dynamics.



(a) $\|K^*\|$



(b) $\kappa(H)$

Figure 1: (a) $\|K^*\|$ as a function of α and γ (b) Condition number of the data matrix $\kappa(H)$ for different N and T

6. Tuning Persistent Excitation

As explained in Section 5, in the *data-collection phase* of the algorithm, persistently exciting (PE) signals were added to control inputs $[u_1, \dots, u_m]$, disturbances $[d_1, \dots, d_q]$, and augmented states $[\bar{X}_{n+1}, \dots, \bar{X}_{2n}]$ while the required information was collected. There are $m + q + n$ required PE signals in this algorithm. As part of this study, we experimented with multiple types of PE signals including white noise, linear chirp, exponential chirp, and sum of sinusoids. We found that a sum of sinusoids PE signal was the most effective at converging to the optimal solution, so the PE signals used in this study are sums of sinusoids in the following format with appropriate amplitudes, a_λ,

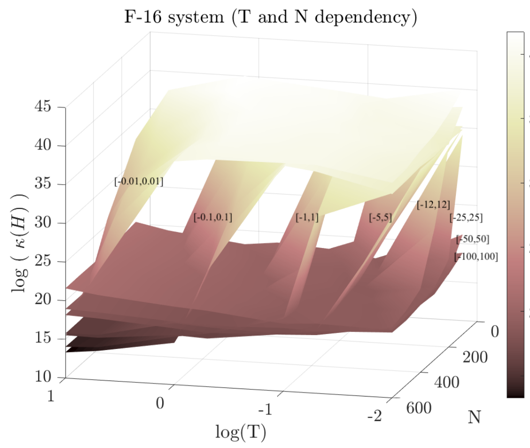
$$\Psi_\lambda(t) = a_\lambda \sum_{i=1}^{\xi=1000} \sin(\omega_{\lambda,i} t), \qquad \omega_{\lambda,i} \in [-\Delta_\lambda, \Delta_\lambda] \tag{33}$$

where $\lambda \in [1, \dots, m, m+1, \dots, m+q, m+q+1, \dots, m+q+n]$. $\omega_{\lambda,i}$ is one of $\xi = 1000$ uniformly distributed random angular frequencies between $-\Delta_\lambda$ and Δ_λ. Both a_λ and Δ_λ are additional hyperparameters that need to be tuned. More terms in the sum of sinusoids PE (larger ξ) make the excitation richer which is why we chose a large value for ξ. In a model-free sense, the amplitudes, a_λ, can be tuned appropriately as long as the system remains stable during the *data-collection phase*. Optimal tuning of the amplitudes was not addressed with this work. However, for fixed amplitudes, we found a strong correlation between the condition number of the data matrix $\kappa(H)$ and the range of the uniformly distributed random angular frequencies when using the same frequency range for all PE signals. We now refer to the frequency range as $[-\Delta, \Delta]$. Since there is no information about the system dynamics in a model-free algorithm, this seems like a reasonable approach. The results presented in Figure 2 show how changes in Δ affect $\kappa(H)$ in this study.

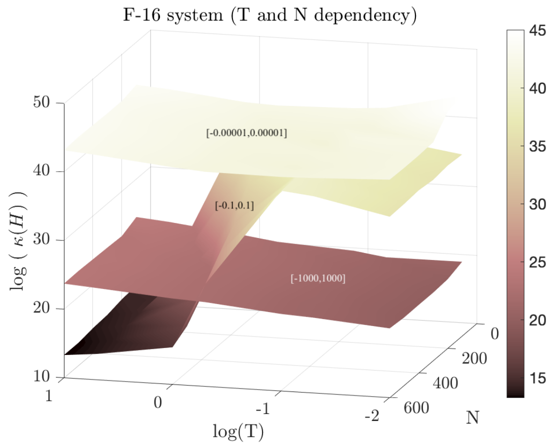
For comparison, in Figure 2 (right), we present our results for the largest ($\Delta = 1000$ rad/s) and smallest ($\Delta = 0.00001$ rad/s) ranges of angular frequencies we could compute along with the one that gives the lowest $\kappa(H)$. The results show that a very small PE (or by extrapolation no PE) will result in high $\kappa(H)$ (ill-conditioned) which makes it impossible for the algorithm to converge to the optimal solution. Conversely, a very wide range like $\Delta = 1000$ rad/s results in a relatively low and constant $\kappa(H)$ for all values of N and T. Subsequently, we found specific range in this example which leads to the lowest $\kappa(H)$ for N and T. In fact, $\Delta = 0.1$ rad/s will result in relatively low $\kappa(H)$ for large T. In other words, N, T, and the range of angular frequencies, Δ, in the PE signals can be tuned via the *Grid Search* described in Section 5 in order to achieve good convergence to the optimal solution. The best values of Δ, N, and T for the example in this study are located in the darkly-shaded regions of Figure 1. Additionally, despite using the same range of frequencies for all PE signals, we found that none of the PE signals individually could account for the lowest condition number. The low condition number comes from the combined influence of each PE signal on the system. Despite having knowledge of the system dynamics for the example in the paper, use of that knowledge in terms of the eigenvalues did not appear to provide any benefit in tuning Δ.

7. Correlation between Condition number and optimal solution

In previous sections, we presented a method for tuning N, T, and PE for a fixed selection of α and γ without requiring a system model. In addition to those hyperparameters, the selection of α and γ is also an important part of the learning algorithm, since they are present in (10). In Section 5, we provided guidelines for choosing α and γ. In this section, we explain this in more detail and show



(a) Gradual development



(b) Minimum and maximum

Figure 2: Condition number of the data matrix $\kappa(H)$ for different N and T when different angular frequency ranges $[-\Delta, \Delta]$ for PE signals are used.

how our model-free *Grid Search* for low condition number $\kappa(H)$ can produce better convergence to the optimal solution.

Using a sum of sinusoids PE signal, we computed condition number $\kappa(H)$, the relative norm of the performance function difference $||J^* - J||/||J^*||$ and the norm of the gain difference $||K^* - K||$ for a range of α and γ values. In these computations, J is the learned performance index (7), J^* is the optimal performance index, K is the learned gain, and K^* is the optimal gain (Figure 1).

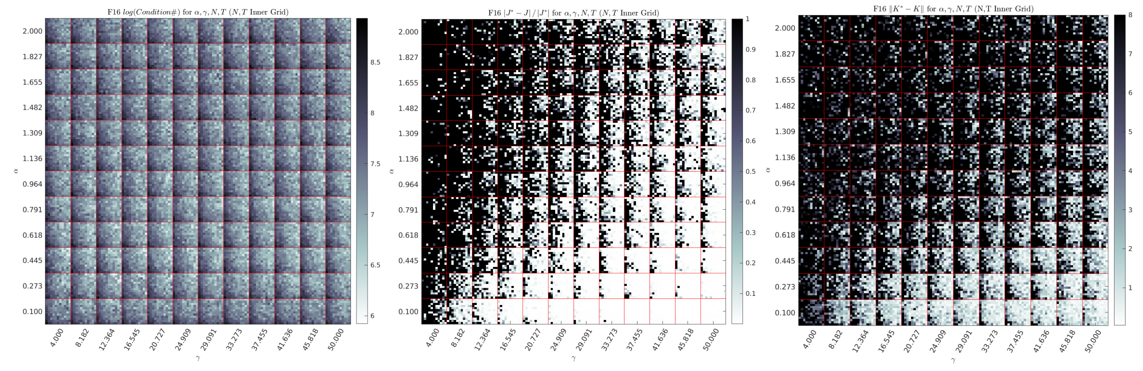
Figure 3 (left) presents condition number of the data matrix $\kappa(H)$ for different N and T as α and γ are varied. In general, for any selection of α and γ, $\kappa(H)$ decreases as N and T increase. In other words, for any selection of α and γ, when we collect more samples with longer sampling interval, we generally obtain a better learned solution. However, note that there may be particular small N and T for any selection of α and γ which can lead to lower $\kappa(H)$ as we see in Figure 1 (right). In fact, as explained in Section 5, $\kappa(H)$ is a great model-free measure to identify these particular N and T which lead to a lighter and faster data collection.

In Figure 3 (middle and right), we depict relative norm of the performance function difference $||J^* - J||/||J^*||$ and norm of the gain difference $||K^* - K||$, respectively, for different N and T when changing α and γ. As α increases, the convergence to the optimal solution degrades.

As can be seen in the plots, a small α with $\gamma > 20$ is a safe selection for any N and T. Also, for each selection of α and γ, as N and T increase, the convergence of the learned solution to the optimal solution gets better. This implies a correlation between low condition number and the ability to converge to the optimal solution.

8. Conclusion

The model-free IRL algorithm discussed in this paper solves the H_∞ optimal tracking problem iteratively by the Ordinary Least Squares method, so the condition number of the data matrix $\kappa(H)$ plays a key role in the convergence of the algorithm. The data matrix is formed after collecting data, so the value of $\kappa(H)$ is computed without any knowledge of the system dynamics. Since the



(a) $\kappa(H)$ (b) $||J^* - J||/||J^*||$ (c) $||K^* - K||$

Figure 3: Condition number of the data matrix $\kappa(H)$, relative norm of performance function difference $||J^* - J||/||J^*||$, and norm of the gain difference $||K^* - K||$ for different N and T as α and γ are varied (N increases to the right, T increases vertically on the inner grid)

choices of number of samples, N, and sampling interval, T, strongly influence the properties of the data matrix, they also strongly influence the convergence properties of the algorithm. We found that, for any selection of α and γ, $\kappa(H)$ generally decreases as N and T increase. In addition, the convergence of the learned solution to the optimal solution generally improves as both N and T increase. In other words, when we collect more samples with longer sampling interval, we generally obtain a better learned solution. Also, for any selection of α and γ, the search for lowest $\kappa(H)$ can lead to particularly small N and T which makes data collection lighter and faster. It is shown that as α increases, the convergence of the learned solution to the optimal solution degrades. For this reason, we recommend starting with a small value for α in order to get the best convergence.

We also found a strong correlation between $\kappa(H)$ and the PE signals in the form of sums of sinusoids. Very small PE (or no PE) results in high $\kappa(H)$ which makes convergence to the optimal solution difficult. In contrast, a very wide range of angular frequencies results in a relatively low and constant $\kappa(H)$. We found specific ranges for the F-16 model which lead to the lowest overall $\kappa(H)$ and we determined none of the PE signals individually could account for it. The combined influence of the PE signals seems to be the determining factor leading to lower condition numbers. This was also true in the case of measurement noise. We found that using measurement noise alone as the PE resulted in higher condition numbers than when we applied the sum of sinusoids PE.

Immediate future work includes investigating the application of our findings to multiple classes of systems where each class is represented by the same equations of motion but with different physical parameters. Examples of system classes could include vehicles with double integrator dynamics, multiple types of fixed-wing aircraft, and multiple types of multirotors.

Acknowledgments

The authors wish to acknowledge the support of their colleagues at the University of Texas at Arlington including Endowed Chair Professor Frank Lewis, Distinguished Professor Yan Wan, and Ph.D. Candidate Patrik Kolarić.

This work was supported in part by the ONR Basic Research Challenge (BRC) program under grant No. N0001421WX02049 and the 2021 Naval Innovative Science and Engineering (NISE) program.

References

James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Marie-Liesse Cauwet, Camille Couprie, Julien Dehos, Pauline Luc, Jeremy Rapin, Morgane Riviere, Fabien Teytaud, Olivier Teytaud, and Nicolas Usunier. Fully parallel hyperparameter search: Reshaped space-filling. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1338–1348. PMLR, 13–18 Jul 2020.

Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. In *XI Metaheuristics International Conference*, pages 1–5, 2015.

James W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997. doi: 10.1137/1.9781611971446.

Alireza Farahmandi, Gary Hewer, Brian Reitz, Katia Estabridis, and Kyriakos G. Vamvoudakis. A model free learning algorithm to control autonomous streams over iot. In *IOT '18: Proceedings of the 8th International Conference on the Internet of Things*, pages 1–4, 10 2018. ISBN 978-1-4503-6564-2. doi: 10.1145/3277593.3277640.

G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, third edition, 1996. ISBN 9780801854149.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, second edition, 2002. ISBN 9780898715217.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017.

Hamidreza Modares, Frank Lewis, and Zhong-Ping Jiang. H_∞ tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. *IEEE transactions on neural networks and learning systems*, 26:2550–2562, 06 2015. doi: 10.1109/TNNLS.2015.2441749.

Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In Hugo Jair Escalante and Katja Hofmann, editors, *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 3–26. PMLR, 06–12 Dec 2021.