Data-Driven Control and Data-Poisoning attacks in Buildings: the KTH Live-In Lab case study

Alessio Russo∗,†, Marco Molinari‡ and Alexandre Proutiere§

Abstract—This work investigates the feasibility of using input-output data-driven control techniques for building control and their susceptibility to data-poisoning techniques. The analysis is performed on a digital replica of the KTH Live-in Lab, a non-linear validated model representing one of the KTH Live-in Lab building testbeds. This work is motivated by recent trends showing a surge of interest in using data-based techniques to control cyber-physical systems. We also analyze the susceptibility of these controllers to data poisoning methods, a particular type of machine learning threat geared towards finding imperceptible attacks that can undermine the performance of the system under consideration. We consider the Virtual Reference Feedback Tuning (VRFT), a popular data-driven control technique, and show its performance on the KTH Live-In Lab digital replica. We then demonstrate how poisoning attacks can be crafted and illustrate the impact of such attacks. Numerical experiments reveal the feasibility of using data-driven control methods for finding efficient control laws. However, a subtle change in the datasets can significantly deteriorate the performance of VRFT.

I. INTRODUCTION

Recent trends have shown a surge of interest in methods that intelligently learn from the data. This trend is motivated by recent successes in using deep-learning based methods for supervised learning tasks. Similarly, data-driven control approaches, which are adaptive control methods, have gathered much attention over the last few decades [11]–[13]. A recent analysis demonstrated that data-driven control techniques are also affected by this particular attack for simple PID-like controllers [14], whilst the case where VRFT is used with non-linear controllers is left unexplored. Similar attacks, conducted at test time, have also been shown to work in the case of systems controlled by means of Reinforcement Learning [15].

Contributions. The objectives of this work are twofold: (1) we first analyze the feasibility of using VRFT for temperature control in buildings. This is validated by using a digital replica of the KTH Live-In Lab testbed [16], a model of the real building set up using IDA Indoor Climate and Energy (IDA ICE) [17], a software used to simulate buildings performance. (2) We then analyze the susceptibility of VRFT to data poisoning attacks, using the IDA ICE environment. We believe this is an important example of how data-driven control laws can be attacked. In buildings, the probability of sensors being hijacked is far from remote, and a malicious agent can use the data in several ways. This data could be used to determine the number of people present in the building or be poisoned to decrease the building’s energy efficiency. Gartner [18] predicts that through 2022 30% of all AI cyberattacks will leverage training-data poisoning, AI model theft, or adversarial samples to attack AI-powered systems. In [19], Microsoft engineers analyzed 28 companies and found out that only 3 of them have the right tools in place to secure their ML systems. This further stresses the importance of studying such problems.

Organization of the paper: §II introduces the notation, the VRFT method, and the KTH Live-in Lab Testbed, a smart residential building located at the KTH campus. In §III, the VRFT method is used to derive a controller that can control the temperature in the KTH Live-in Lab testbed’s model. Finally, in §IV, the data poisoning attack from [14] is presented and applied to the VRFT method introduced in §II.

II. BACKGROUND AND PRELIMINARIES

A. Notation

We consider discrete-time models, indexed by \( t \in \mathbb{N}_0 \), and we will indicate by \([N]\) the sequence of integers from 0 to \( N \).
We denote by $z$ the one-step forward shift operator and by $\mathcal{H}_2$ the Hardy space of complex functions which are analytic in $|z| < 1$ for $z \in \mathbb{C}$. For a vector $x \in \mathbb{R}^n$ and a function $f : \mathbb{R}^n \to \mathbb{R}$, we denote by $\nabla_x f(x)$ the $n$-dimensional vector of partial derivatives, where each element is $\partial f / \partial x_i$. We will describe a linear time-invariant system in the following way for $t \in \mathbb{N}_0$:

$$
x_{t+1} = Ax_t + Bu_t, \quad x_0 \in X_0 \quad (1)
$$

$$
y_t = Cx_t + Du_t, \quad (2)
$$

where $x_t \in \mathbb{R}^n$ is the state of the system, $u_t \in \mathbb{R}$ is the exogenous input, $y_t \in \mathbb{R}$ is a vector of measurements and $X_0$ is a closed-convex subset of $\mathbb{R}^n$. We can equivalently use transfer function notations and denote the input-output relationship using the transfer function $G : y_t = G(z)u_t$, with $G(z) = C(zI - A)^{-1}B + D$. We also denote the multiplication of two transfer functions $G(z)$ and $L(z)$ by $GL(z)$ (similarly for the sum). Finally, we will denote by $X_T = \{x_0, \ldots, x_T\}^T$ a matrix of dimensions $(T + 1) \times n$ containing a collection of state measurements of the system, for $T \in \mathbb{N}_0$. Similarly, we also define $U_T$ and $Y_T$.

**B. Virtual Reference Feedback Tuning**

In the following, we will denote by $\mathcal{D}_N = (U_N, Y_N)$ the data available to the learner that comes from experiments on the plant, with $N > 1$. This data will be used to learn the control law, and it is usually assumed to have been taken in open-loop conditions. In VRFT [1], the design requirements are encapsulated into a reference model $M_r(z)$ that captures the desired closed-loop behavior from $r_t$ to $y_t$, where $r_t \in \mathbb{R}$ is the reference signal. We assume that $M_r$ satisfies some realizability assumptions, such as being a proper stable transfer function.

In VRFT we wish to find a controller $K_\theta(z)$, parametrized by a vector of $n_k \in \mathbb{N}$ parameters $\theta \in \mathbb{R}^{n_k}$, that reduces the distance between the reference model and the closed-loop system in the $\mathcal{H}_2$ norm sense. Define $\Delta_\theta(z) = M_r(z) - [(I + GK_\theta)^{-1}GK_\theta](z)$, then the objective function to minimize is chosen as

$$
J_{\mathcal{MR}}(\theta) = \|M_r(z) - [(I + GK_\theta)^{-1}GK_\theta](z)\|_2 \quad (3)
$$

$$
= \frac{1}{2\pi} \int_{-\pi}^{\pi} \text{Tr} \left[ \Delta_\theta(e^{j\omega})\Delta_\theta^\top(e^{-j\omega}) \right] d\omega. \quad (4)
$$

One can immediately observe that $J_{\mathcal{MR}}(\theta)$ is non-convex in $\theta$. To address this difficulty, the following assumption [1], [3], [6] is often used:

**Assumption 1.** The sensitivity function $I - M_r(z)$ is close (in the $\mathcal{H}_2$ norm sense) to the actual sensitivity function $(I + GK_\theta)^{-1}(z)$ at the minimizer $\hat{\theta}$ of (3).

This assumption allows us to instead consider the following criterion

$$
\hat{J}_{\mathcal{MR}}(\theta) = \|M_r(z) - [(I - M_r)GK_\theta](z)\|_2^2. \quad (5)
$$

Minimizing (5) can be cast [1] as a problem that involves minimizing the difference between the input signal $u_t$, injected during the experiments, and the control signal $K_\theta(z)e_t$, computed using the virtual error signal $e_t$. The latter is defined as $e_t = r_t - y_t = (M_r^{-1}(z) - 1)y_t$, where $r_t = M_r^{-1}(z)y_t$ is the virtual reference signal, computed using the reference model $M_r(z)$. Unfortunately, this minimization leads to a biased estimate of the minimizer if the controller for which the cost function to zero is not in the controller set. To address this problem, it is possible to introduce a filter $L(z)$ that pre-filters the data $\mathcal{D}_N$, as shown in [1]. Then, assuming that the data has already been pre-filtered using $L(z)$, we can introduce the criterion that is actually minimized instead of (5):

$$
J_{\mathcal{VR}}(\theta, \mathcal{D}_N) = \frac{1}{N+1} \sum_{t=0}^{N} \|u_t - K_\theta(z)\|_2^2. \quad (6)
$$

It is possible to prove [1] that for stationary and ergodic signals $\{y_t\}$ and $\{u_t\}$ the following asymptotic result holds:

$$
\lim_{N \to \infty} J_{\mathcal{VR}}(\theta, \mathcal{D}_N) = J_{\mathcal{VR}}(\theta),
$$

where

$$
J_{\mathcal{VR}}(\theta) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \text{Tr} \left[ \Delta_\theta(e^{j\omega})\Phi_u(\omega)\Delta_\theta^\top(e^{-j\omega}) \right] d\omega
$$

$$
\Delta_\theta(z) := I - [K_\theta(I - M_r)G](z),
$$

with $\Phi_u$ being the power spectral density of $u_t$. Let $K^*$ denote the minimizer over all possible transfer functions $K(z)$ of $\|M_r(z) - [(I - M_r)GK](z)\|_2^2$. If $K^* \in \{K_\theta : \theta \in \mathbb{R}^{n_k}\}$, then $K^*$ is also the minimizer of (6). Otherwise, one can properly choose a filter $L(z)$ to filter the experimental data so that the minimizer of (6) and (5) still coincide (refer to [1] for details). In the following we assume the control set is linearly parametrized in terms of a basis of transfer functions $\beta(z) = [\beta_1(z) \ldots \beta_n(z)]$ of dimension $n_k$.

**Assumption 2.** The control law $K$ is represented by an LTI system $K_\theta(z)$ that is linearly parametrized by $\theta \in \mathbb{R}^{n_k}$, and we will write $K_\theta(z) = \beta^\top(z)\theta$, with $\beta(z)$ being a vector of linear discrete-time transfer functions of dimension $n_k$.

Assumption 2 includes different types of control law, such as PID, and can be relaxed to other types of models, including neural networks [5], [20].

**C. KTH Live-In Lab Testbed and IDA ICE**

The Live-In Lab Testbed KTH [16] (see Fig. (1)) is located in one of Einar Mattsson’s three plus-energy buildings (see Fig. (2)) in the KTH Main Campus, in Stockholm. The Testbed KTH premises feature a total of approximately 305 m²² distributed over 120 m² of living space, 150 m² of technical space, and an office of 20 m². The modeled living space features four apartments; each apartment has a separate living room/bedroom and a bathroom and shares the kitchen as a common space. Space heating is provided via ventilation. The testbed, which is part of the larger Live-In Lab testbed platform, is designed to be energetically independent, with dedicated electricity generation systems through PV panels, heat generation (ground source heat pumps), and storage (electricity and heat) systems. Sensors are extensively used to monitor and control the indoor climate, to improve energy...
efficiency, study user behavior, and to improve control and fault detection strategies.

In this paper, a digital replica of the testbed that focuses on one apartment was created using the IDA ICE software [17]. IDA ICE is a state-of-the-art dynamic simulation software for energy and comfort in buildings. In order to assess the control laws that we derived, we set up a co-simulation environment that allowed IDA ICE and a Python script to communicate and exchange data through APIs available in IDA ICE.

### III. VRFT Method and Temperature Control

In this section, we shortly describe how the VRFT method is applied to derive a controller. (1) We briefly illustrate the HVAC (Heating, ventilation, and air conditioning) architecture of the testbed; (2) outline the usage of VRFT; (3) conclude with a performance analysis of the derived controllers.

**A. Method and experiments**

**HVAC architecture.** Fig. (3) shows a model of the HVAC architecture of the Live-In Lab Testbed KTH. VRFT is applied to the ventilation control unit that regulates the amount of airflow supplied from the central Air Handling Unit (AHU) to the various apartments in the buildings. Measurements coming from the apartment include the temperature $T(t)$ and $CO_2(t)$ readings, sampled every 540 seconds (9 minutes).

**Experiment setup.** The first step involves designing an experiment that permits the user to gather informative data from the plant. The data is then used to compute a control law by means of VRFT. We decided to gather data from an empty apartment during winter months, in order to test temperature control. To emulate real weather conditions, we used weather data from the local station in Bromma. The simulations were conducted in open-loop, and the input data $u_t$ was chosen to be distributed according to a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. Since the amount of airflow can be expressed as a percentage, the input data was clipped between 0 and 1.

However, due to this saturation effect, one needs to pay extra attention while designing the experiments. To that aim, we designed two scenarios: scenario (A), where the mean of the control law is $\mu = 0.5$ and the standard deviation is $\sigma = 1/6$; scenario (B), where the parameters are set to $\mu = 0.5$ and $\sigma = 1$. Scenario (B) represents the case where the user does not take into consideration the saturation effect. In contrast, in scenario (A) we are guaranteed that with 99% probability the control action $u_t$ belongs to $[0, 1]$ (at the cost of having a crest factor of 3). Lastly, we considered the amount of data gathered for the training process. We decided to consider two cases: one where $N = 100$ data points are used (roughly 10 hours of data with a sampling time of 540 seconds), and the other one where $N = 1000$ data points are used (that is 150 hours of data). Finally, we generated 50 sets of simulations for each scenario to average out results.

![Fig. 3: HVAC architecture of the KTH Live-In Lab testbed.](image)

**Reference model and control law.** In VRFT, the user specifies the closed-loop system’s design requirements by choosing a specific reference model $M_r(z)$. This model, together with the data gathered during the experiments, is used to derive a control law $K_\theta(z)$. We opted for a simple reference model, and assumed that the closed-loop response of the system can be well represented by a second-order system. In practice, we assumed that it takes approximately one hour for the heating system in consideration to increase the temperature from 15° to 21° degrees Celsius in the apartment. Consequently, we have chosen a reference model of the type

$$M_r(z) = \frac{(1 - \lambda)^2}{z^2 - 2\lambda z + \lambda^2},$$

where $\lambda = e^{-T_s \omega_0}$ with $\omega_0 = 0.002$ [rad/s] and $T_s = 540$ [s]. Fig. (5) shows the response of $M_r(z)$ to a step signal (with amplitude 21, starting from an initial temperature of roughly 15° [C]), and its Bode plot. All the data was pre-filtered using a filter $L(z) = (1 - M_r(z))M_r(z)$ (as explained...
in §II; or see [1] for more details). Finally, we chose $K_\theta(z)$ to be a PID controller, of the form

$$K_\theta(z) = \beta(z) = \sum_{k=1}^{3} \theta_k z^{-k+2}.$$  

This is one of the simplest, and most widely used controller. Moreover, it has a linear structure, which makes it feasible to use with VRFT. Future work involves the analysis of more complex controllers, such as neural networks.

**B. Performance validation and results**

**Validation of the controllers.** We conducted 50 different simulations for each scenario, for a total of 200 simulations. The performance of each controller $K_{\theta(i)}(z)$, where $\theta(i)$ is the parameter vector of the $i$-th controller (one for each simulation, i.e., $i = 1, \ldots, 200$), was validated over a temporal period of 2 weeks (2240 data points). During those 2 weeks, the apartment was occupied by one person (according to the occupancy profile shown in Fig. (6)).

**Performance criteria.** The performance of a controller $K_{\theta(i)}(z)$ was evaluated on the basis of two criteria: (1) the RMSE of the temperature signal $e_{\text{RMSE}} = \sqrt{\frac{1}{N} \sum_{t=1}^{N} (\theta(T(t)) - \tau(s))^2}$, where $T(t)$ is the temperature of the living room, and $\tau(t)$ is a constant reference temperature of $21^\circ$C [C]; (2) the average power spectral density of $T(t)$: $e_{\text{PSD}} = \frac{1}{(2\pi f_s)} \int_{0}^{f_s} S_T(f) df$, where $1/(2T_s)$ is the Nyquist frequency and $S_T(f)$ is the power spectral density (PSD) of the temperature $T(t)$ (computed using Welch’s method).

**Results and discussion.** A summary of the results are shown in Table (I). We decided to classify $K_{\theta(i)}(z)$ to be a “good” controller if the following ellipse condition $e_{\text{RMSE}}^2 + (e_{\text{PSD}})^2 \leq 1$ is satisfied. This condition guarantees that $K_{\theta(i)}(z)$ satisfies good tracking performance, and small oscillations. Overall, we found no major difference in performance by using 100 or 1000 data points for Scenario A, whilst there is a clear difference in using 100 or 1000 datapoints for Scenario B. In the latter case, using fewer points may result in controllers with poor performance, as indicated by the results. Surprisingly, using 1000 datapoints for Scenario B results in high performance controllers. Nonetheless, the difference with controllers found in scenario A is minimal.

| Scenario | $N = 100$ | $N = 1000$ |
|----------|-----------|------------|
| Scenario A | $36 \pm 0.2$ | $21\pm 0.3$ | $17.9 \pm 0.25$ | $100\%$ |
| Scenario B | $36 \pm 0.2$ | $70 \pm 0.42$ | $25.82 \pm 0.45$ | $84\%$ |
| Scenario A | $39 \pm 0.1$ | $12 \pm 0.1$ | $15 \pm 0.44$ | $100\%$ |
| Scenario B | $33 \pm 0.1$ | $70 \pm 0.1$ | $5.95 \pm 0.4$ | $100\%$ |

**TABLE I:** For each scenario are shown the average value and confidence level at 95% computed over 50 simulations. Percentage of good controllers indicates the proportion of controllers that falls inside the ellipse $e_{\text{RMSE}}^2 + (e_{\text{PSD}})^2 = 1$.

**IV. DATA POISONING OF VRFT**

In this section, we first present the data poisoning attack, introduced in [14], which inherits the main characteristics of the attack formulated in [11]. We then apply the poisoning attack to the data used in the previous section to compute VRFT, and conclude with a performance analysis of the poisoned controllers.
A. Attack framework and setup

Attack formulation. We now assume that a malicious agent has access to the experimental data \( \mathcal{D}_N \), and knows the reference model \( M_r(z) \) used by VRFT to identify a controller. The goal of the malicious agent is to degrade the performance of the resulting closed-loop system by subtly changing the dataset \( \mathcal{D}_N \).

We denote the malicious signal on the actuators by \( a_u \in \mathbb{R}^m \), and respectively by \( a_y \in \mathbb{R}^p \), the attack signals on the sensors at time \( t \). The new input and output data points in the dataset at time \( t \) are, respectively, \( u'_t = u_t + a_u \), and \( y'_t = y_t + a_y \). We denote the corrupted dataset by \( \mathcal{D}'_N = (U'_N, Y'_N) \) where \( U'_N = [u'_0 \quad u'_1 \quad ... \quad u'_N]^\top \) and similarly \( Y'_N = [y'_0 \quad y'_1 \quad ... \quad y'_N]^\top \). We focus our attention on the maxmin attack, introduced in [14], which is casted as a bilevel optimization problem

\[
\begin{align*}
\max_{U_N,Y_N} & \quad J_{VR}(\hat{\theta}', \mathcal{D}_N) \\
\text{s.t.} & \quad \hat{\theta}' \in \arg \min_{\theta} J_{VR}(\theta, \mathcal{D}'_N), \quad (7)
\end{align*}
\]

where the constraints limit the amount of change applied to the dataset \( \mathcal{D}_N \). In a maxmin attack, the malicious agent aims at maximizing the learner’s loss. By choosing this cost function, the malicious agent is implicitly maximizing the residual error \( \|M_r(z) - [(I - M_r)G]K_{\theta}(z)\|_2 \) (as \( N \to \infty \)). Despite the simplicity of this criterion, the resulting closed-loop system may remain stable, or just be slightly affected by the attack. One can formulate alternative criteria, but for the sake of simplicity, we will restrict our analysis to the maxmin attack.

Moreover, we want to highlight a few differences compared to classical data poisoning in supervised learning models. First, there is no label for the data, which implies that we cannot merely maximize the probability of classification error. Second, the problem involves two sets of data, the input \( U_N \), and the output data \( Y_N \). Since the dependency of the solution may depend in a complicated way on \( U_N \) and \( Y_N \), the problem is harder.

Convexity. It can be shown that the optimization problem (7) is convex in \( U' \) for a fixed \( Y' \). Therefore, the maximum over \( U' \), for some \( Y' \), is attained on some extremal point of the feasible set. To find the optimal attack vector on the input, one can use, for example, disciplined convex-concave programming (DCCP) [21]. However, convexity with respect to \( Y' \) does not hold. Therefore, we decided to use gradient ascent methods to compute a solution.

Algorithm and setup. Based on the previous discussion, we use Alg. (1) to approximately solve problem (7). We first perform a change of variables \( A_u = U'_N - U_N \) and \( A_y = Y'_N - Y_N \), and then solve the problem in the new variables \((A_u, A_y)\). The algorithm first solves (7) in the input variable \( A_u \) using DCCP, and then in the output variable \( A_y \) using PGA (Projected Gradient Ascent). For both DCCP and PGA, we pick uniformly at random 20 initial points at every iteration. The algorithm stops whenever the increase between one iteration and the other is not greater than a fixed user-chosen value \( \eta > 0 \).

Algorithm 1: Max-min attack algorithm

Input: Dataset \( \mathcal{D}_N = (U_N,Y_N) \); parameters \( \delta_u, \delta_y, \eta \)
Output: Poisoned dataset \( \mathcal{D}'_N \)
1. \( i \leftarrow 0 \), \( (A_u^{(i)}, A_y^{(i)}) \leftarrow (0,0) \) \( \triangleright \) Initialize algorithm
2. \( \hat{\theta}^{(i)} \leftarrow \arg \min_{\theta} J_{VR}(\theta, U_N + A_u^{(i)}, Y_N + A_y^{(i)}) \)
3. do
4. \( A_u^{(i+1)} \leftarrow \text{solve (7) in } A_u \text{ using DCCP} \)
5. \( A_y^{(i+1)} \leftarrow \text{solve (7) in } A_y \text{ (using } A_u^{(i+1)} \text{) with PGA} \)
6. \( \hat{\theta}^{(i+1)} \leftarrow \arg \min_{\theta} J_{VR}(\theta, U_N + A_u^{(i+1)}, Y_N + A_y^{(i+1)}) \)
7. \( i \leftarrow i + 1 \)
8. while \( J_{VR}(\hat{\theta}^{(i)}, U_N, Y_N) - J_{VR}(\hat{\theta}^{(i-1)}, U_N, Y_N) > \eta \)
9. return \( \mathcal{D}'_N = (U_N + A_u^{(i)}, Y_N + A_y^{(i)}) \)

B. Performance and results

Setup. As in the previous section, we consider 4 configurations: Scenario A with 100/1000 data points, and similarly Scenario B. For each configuration, we chose \( \delta_u \) and \( \delta_y \) in (7) as \( \delta_u = \varepsilon_u \|U_N\|_2 \) and \( \delta_y = \varepsilon_y \|U_N\|_2 \), where \( \varepsilon_u, \varepsilon_y \) are positive parameters in \([0, 1]\) that we used as control knobs to vary the amount of change in \( \mathcal{D}_N \). For simplicity, we also assumed that the data has already been pre-filtered using the filter \( L(z) \).

Results. The main result is shown in Fig. (7), whilst in Fig. (8) is shown an example of poisoned dataset for Scenario A.

![Example of poisoned (pre-filtered) data for scenario A with 1000 data points, and \((\varepsilon_u, \varepsilon_y) = (0.1, 0.2)\).](image-url)
Poor tracking
More oscillations
Scenario A – 1000 Data Points -
Scenario B – 1000 Data Points -

Fig. 7: Results for the data poisoning attack. Each point on the left plots represents the average across 50 simulations for a specific set of values $(c_u, c_y)$, displayed on the top of each point (also the unpoisoned cases are depicted in the plots).

to the attack. However, Scenario A, for a large number of data points, is also significantly affected by the attack, while using a low number of data points seems to improve robustness. Unfortunately, as depicted in Fig. (8), minimal changes lead to a substantial performance degradation, as shown in the bottom right plot in Fig. (7). These results stress the importance of designing experiments properly, and make sure that the gathered data is secured.

V. CONCLUSION

In this work, we have shown the feasibility of VRFT, an input-output data-driven method, for comfort control in buildings, namely temperature control, and analyzed the impact of the maximin data poisoning attack. VRFT has been validated on a digital replica of the KTH Live-In Lab, modeled using IDA-ICE, showing good performance and small tracking error. We then analyzed the impact of data poisoning attacks, which revealed that small changes in the dataset could disrupt the controller’s performance. Results also indicated that smaller datasets are more robust to data poisoning attacks, while datasets naively constructed are more susceptible to the attack, resulting in substantial performance degradation. This stresses the importance of securing the data used to derive the control law.

ACKNOWLEDGEMENTS

This work was supported by the Swedish Foundation for Strategic Research through the CLAS project (grant RIT17-0046).

REFERENCES

[1] M. C. Campi, A. Lecchini, and S. M. Savaresi, “Virtual reference feedback tuning: a direct method for the design of feedback controllers,” Automatica, vol. 38, no. 8, pp. 1337–1346, 2002.

[2] H. Hjalmarsson, M. Gevers, S. Gunnarsson, and O. Lequin, “Iterative feedback tuning: theory and applications,” IEEE control systems magazine, vol. 18, no. 4, pp. 26–41, 1998.

[3] A. Karimi, K. Van Heusden, and D. Bonvin, “Non-iterative data-driven controller tuning using the correlation approach,” in 2007 European Control Conference (ECC). IEEE, 2007, pp. 5189–5195.

[4] C. De Persis and P. Tesi, “Formulas for data-driven control: Stabilization, optimality, and robustness,” IEEE Transactions on Automatic Control, vol. 65, no. 3, pp. 909–924, 2019.

[5] A. Esparza, A. Sala, and P. Albertos, “Neural networks in virtual reference tuning,” Engineering Applications of Artificial Intelligence, vol. 24, no. 6, pp. 983–995, 2011.

[6] S. Formentin, K. Van Heusden, and A. Karimi, “A comparison of model-based and data-driven controller tuning,” International Journal of Adaptive Control and Signal Processing, vol. 28, no. 10, 2014.

[7] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, “A note on persistency of excitation,” Systems & Control Letters, vol. 54, no. 4, pp. 325–329, 2005.

[8] J. D. Rejas, X. Flores-Alsina, U. Jeppsson, and R. Vilanova, “Application of multivariate virtual reference feedback tuning for wastewater treatment plant control,” Control Engineering Practice, vol. 20, no. 5, pp. 499–510, 2012.

[9] D. Invernizzi, P. Panizza, F. Riccardi, S. Formentin, and M. Lovera, “Data-driven attitude control law of a variable-pitch quadrotor: a comparison study,” IFAC-PapersOnLine, vol. 49, no. 17, 2016.

[10] Y. Li, J. Shen, and K. Y. Lee, “Data-driven nonlinear control of a solid oxide fuel cell system,” IFAC Proceedings Volumes, vol. 44, no. 1, pp. 14778–14783, 2011.

[11] B. Biggio, B. Nelson, and P. Laskov, “Poisoning attacks against support vector machines,” in Proceedings of the 29th International Conference on International Conference on International Conference on Machine Learning, ser. ICML’12. Madison, WI, USA: Omnipress, 2012, p. 14671474.

[12] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, “Manipulating machine learning: Poisoning attacks and countermeasures for regression learning,” in 2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018, pp. 19–35.

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” arXiv preprint arXiv:1412.6572, 2014.

[14] A. Russo and A. Proutiere, “Poisoning attacks against data-driven control methods,” in 2021 American Control Conference (ACC), May 26-28, 2021, New Orleans, USA. IEEE, 2021.

[15] ——, “Towards optimal attacks on reinforcement learning policies,” in 2021 American Control Conference (ACC), May 26-28, 2021, New Orleans, USA. IEEE, 2021.

[16] KTH Royal Institute of Technology, “KTH Live-In Lab.” [Online]. Available: https://www.liveinlab.kth.se

[17] EUQUA Simulation AB, “IDA Indoor Climate and Energy (IDA-ICE),” 2020, version 5.0. [Online]. Available: https://www.equa.se/en/ida-ice

[18] B. Burke, D. Cearley, N. Jones, D. Smith, A. Chandrasekaran, C. Lu, and K. Panetta, “Gartner top 10 strategic technology trends for 2020-smarter with gartner,” 2019.

[19] R. S. S. Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia, “Adversarial machine learning–industry perspectives,” arXiv:2002.05648, 2020.

[20] A. Sala and A. Esparza, “Extensions to virtual reference feedback tuning: A direct method for the design of feedback controllers,” Automatica, vol. 41, no. 8, pp. 1473–1476, 2005.

[21] X. Shen, S. Diamond, Y. Gao, and S. Boyd, “Disciplined convex-concave programming,” in 2016 IEEE 55th Conference on Decision and Control (CDC). IEEE, 2016, pp. 1009–1014.