MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Adaptive Velocity Estimators for Learning Control

Nikovski, Daniel; Yerazunis, William S.

TR2024-088 July 02, 2024

Abstract

The paper proposes a method for learning velocity estimators, in the form of finite impulse response (FIR) filters, from data collected from a system equipped with quantizing position encoders that is to be controlled by means of a full state feedback controller making use of the velocity estimates. The resulting estimators are tailored to the proper- ties of the controlled system and show empirically superior performance in comparison with commonly used baseline velocity estimators, both in terms of velocity estimation error as well as in terms of reduced regulation cost when tested on control problems. The proposed adaptive estimators are resistant to overfitting the training data, are easy to implement on embedded controller devices, and can be used in conjunction with various learning control methods.

International Conference on Control, Decision and Information Technologies (CoDIT) 2024

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Adaptive Velocity Estimators for Learning Control

Daniel Nikovski and William Yerazunis†

***Abstract*—**

The paper proposes a method for learning velocity estimators, in the form of finite impulse response (FIR) filters, from data collected from a system equipped with quantizing position encoders that is to be controlled by means of a full state feedback controller making use of the velocity estimates. The resulting estimators are tailored to the properties of the controlled system and show empirically superior performance in comparison with commonly used baseline velocity estimators, both in terms of velocity estimation error as well as in terms of reduced regulation cost when tested on control problems. The proposed adaptive estimators are resistant to overfitting the training data, are easy to implement on embedded controller devices, and can be used in conjunction with various learning control methods.

***Index Terms*— Learning control, state estimation**

I. Introduction

Velocity estimation plays an important role in the control of many mechanical systems whose state is described by both position variables (angles or distances) as well as velocities (angular or linear). More often than not, dedicated velocity sensors (tachometers) are not available, and the velocities need to be estimated from position sensors, most commonly rotary encoders. Encoders introduce quantization noise which depends on the resolution of the encoder. This noise, when added to the actual velocity signal, ends up creating a disturbance in the control signal that can significantly worsen the performance of the controller and even render it unstable. As the cost of encoders increases sharply with their resolution, there is a strong economic motivation to develop effective velocity estimation methods that can mitigate the effect of quantization noise on the estimated velocities. In contrast to spatial resolution, increasing the temporal resolution of the estimation process (the sampling rate of the encoder) is much more economical due to the ever increasing computational capabilities of modern microcontrollers. The question then arises whether higher temporal resolution can be leveraged to compensate for limited spatial resolution of velocity estimation. The answer to this question is, though, not straightforward, because directly increasing the sampling rate without changing the velocity estimation scheme will actually amplify the estimation error due to quantization, and not reduce it. Consequently, it becomes important to devise new methods for improved velocity estimation that take advantage of higher sampling rates, and this paper proposes one such method.

If a state-space model of the controlled plant is available, a suitable velocity observer can be designed by leveraging the model. However, in learning control applications, such a state-space model is not available; rather, it is the objective of the learning algorithm to learn such a model from observations (for model-based methods) or learn directly a control law (also called a policy in the field of reinforcement learning (RL), for model-free methods) that maps the measured or estimated state variables to the control variables. Such learning algorithms must use *apriori* velocity estimators that are not informed by the plant's model.

This raises the question of which velocity estimator method is optimal and advantageous to use, and a significant amount of research has been performed on the performance of various velocity estimators. As velocity is the first derivative of position, this is an instance of the more general problem of differentiating a digital signal $x[k] = x(t_k)$ sampled at discrete moments in time $t[k]$, $k = 0, \ldots, N-1$ and corrupted by general noise. The simplest and most fundamental velocity estimator is the first-order backward difference estimator (BDE), defined as the ratio of the difference between consecutive samples of the signal and the time elapsed between them:

$$\hat{v}[k]^{BDE} \triangleq \frac{\Delta x[k]}{T[k]}, \tag{1}$$

where $\Delta x[k] \triangleq x[k] - x[k-1]$ and $T[k] \triangleq t[k] - t[k-1]$. When the true positions $x^{(true)}[k]$ have been corrupted by independent and identically distributed (i.i.d.) quantization noise $\epsilon[k]$, such that $x[k] = x^{(true)}[k] + \epsilon[k]$, the difference $\Delta\epsilon[k] \triangleq \epsilon[k] - \epsilon[k-1]$ is propagated into the velocity estimate as $\Delta\epsilon[k]/T[k]$, manifesting itself as velocity estimation error. For control applications, the sampling period $T[k]$ is often the inverse of the control rate F, meaning that that rate effectively amplifies the quantization noise of the encoder. Furthermore, if, for example, a linear feedback controller multiplies the velocity estimate by a velocity gain k_v, the resulting additive disturbance to the control signal will be equal to $k_v F \Delta\epsilon[k]$. Depending on the gain, the control rate, and the quantization error, this disturbance might be very large and completely dwarf the other components of the control signal, especially at low velocities, rendering the control system unstable.

Given the deleterious effect of higher sampling rates on the velocity estimation error and the resulting induced disturbance in the control signal, the simplest solution would

†Both authors are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA 02139 {nikovski,yerazunis}@merl.com

be to resort to slower sampling rates. However, this would ignore information in the encoder position signal that might be instrumental in more accurate estimation. A much better solution might be to try to make better use of multiple position readings sampled at a high control rate. To this end, a number of more advanced velocity estimators have been proposed [1]. A Taylor series expansion (TSE) of the velocity signal leads to a series of estimators of arbitrary order. Similarly, the BDE estimators can be extended to an arbitrary order. The first- and second-order TSE and BDE estimators are the same, and differences appear only in the third- and higher-order terms. In practical applications, at most third-order estimators are usually used. Both TSE and BDE estimators can be implemented as causal linear finite impulse response (FIR) filters, either on the raw position measurement $x[k]$ or on the differences $\Delta x[k]$. A different, and computationally much more intensive approach is to fit a low-order polynomial to the latest readings using least-squares fitting (LSF), differentiate it analytically, and evaluate the derivative at the most recent point in time to obtain a velocity estimate [1], [2]. Variants of these estimators exist when the encoders are not sampled at fixed time steps, but produce a sample only when their position reading changes by a fixed amount, typically one tick of the encoder.

Looking back at the long research into optimal velocity estimators, it becomes clear that no single estimator has proved to be best for all applications and systems. This should come as no surprise, as the performance of these estimators, and most notably their ability to suppress quantization noise and serve in feedback control applications, depends critically on the properties of the system whose velocity is being measured. The most important among these properties is the bandwidth of the system, which determines the frequency range within which the measured velocities are going to lie. For a system of zero bandwidth, that is, always moving at a constant velocity, the best estimate of that velocity would be simply the average velocity $\bar{v}[k] = (x[k]-x[0])/(t[k]-t[0])$ since the start; by dividing the difference $(\epsilon[k]-\epsilon[0])$ between the i.i.d. quantization errors in the first and last readings by the largest possible time interval, $t[k]-t[0]$, the quantization noise would be suppressed most effectively. This operation is also equivalent to passing the position differences $\Delta x[i]$, $i = 1, \ldots, k$ through an FIR filter of order k with coefficients all equal to $1/k$. However, this kind of very high-order causal filter will cause a huge phase shift in the estimated signal for all non-zero frequencies, and because it will act in sequence with the feedback controller if used for control purposes, it will likely affect its phase margin very negatively. So, for systems exhibiting high-frequency velocities, there might be no other choice but to use a velocity estimator of the lowest possible order. That is, various velocity estimators provide a different trade-off between suppression of quantization noise and phase lag, and which one is better for a particular system depends on the bandwidth of the system, which is typically unknown in learning control applications.

Based on this realization, in this paper we propose a method for adaptively constructing a velocity estimator that is tailored to the properties of a target system to which learning control is being applied. Similar to TSE and BDE estimators, the resulting estimator is an FIR filter that acts on the measured encoder position differences, but unlike these estimators, its order and coefficients are learned from data, using standard machine learning (ML) methodology. In the spirit of learning control methods, it applies ML technology beyond its usual uses for system identification and policy learning, to the problem of velocity estimation. Section II describes the proposed method, Section III details the verification task, Section IV presents empirical results, and Section V proposes directions for future work and concludes.

II. Learning Velocity Estimators

The overall idea of the method is to excite the target system by means of a suitable excitation policy, collect encoder data, and learn a suitable estimator in the form of an FIR filter. Later, the filter is used to estimate velocities while learning a model of the system for the purpose of model-based controller design, or direct determination of a control policy based on position and velocity readings. Finally, the same filter is used when deploying the learned control policy. The method is also compatible with control designs that are based on an approximate state-space model of the plant, whose accuracy is sufficient for a controller design with acceptable performance, but insufficient for designing a model-based observer of velocity.

The method is based on the realization that many velocity estimators known from the literature, such as the TSE and BDE estimators, are essentially weighted moving averages of the position differences $\Delta x[k]$ of the position data, as provided by position encoders. The weights of the moving average can be represented as the impulse response h_i, $i = 0, n-1$ of an FIR filter of order n, where n is the order of the velocity estimator, adopting the terminology of [1], and the velocity estimate is the result of convolving the impulse response with the signal of position differences:

$$\hat{v}_k = h_0\Delta x[k] + \cdots + h_{n-1}\Delta x[k-n+1] = \sum_{i=0}^{n-1} h_i \Delta x[k-i] \tag{2}$$

The only difference between TSE and BDE estimators is in what kind of impulse response $h = [h_0, h_1, \ldots, h_{n-1}]$ they use. For example, the first order TSE(1) and BDE(1) filters use, trivially, $h = [1]$. Their second-order variants, TSE(2) and BDE(2), use the same convolution kernel, $h = [1.5, -0.5]$. Differences between TSE and BDE gradually appear in the higher-order filters. However, in practical applications, it is not clear why one kernel should be preferred to another of the same, or for that matter, a different order.

We propose to make this decision based on collected data, and moreover, use that data to find the actual optimal kernel h that is best for the system from which the data was collected, in the expectation that this kernel will generalize to novel data from the same system, as is customary in the field of ML. To this end, we formulate the following

machine learning problem. Given a sequence of position differences $\Delta x[k]$, $k = 1, N-1$, best off-line estimates $\hat{v}^{(o)}[k]$ of the velocities at the same time moments, and a desired FIR filter order n, find the values of the impulse response $h = [h_0, h_1, \ldots, h_{n-1}]$ that minimize the mean-squared prediction error of the velocities:

$$h^{(LSE)} = \arg\min_{h} \sum_{k=n+1}^{N-1} \left(\hat{v}^{(o)}[k] - \sum_{i=0}^{n-1} h_i \Delta x[k-i] \right)^2 \tag{3}$$

It can immediately be recognized that this is a least-squares estimation (LSE) problem that can be solved in many ways, for example by forming the time-lag matrix X of dimensions $N-n+1 \times n$, such that $X_{ij} = \Delta x[i-j+n]$, the column vector v of dimension $N-n+1$, such that $v_i = \hat{v}^{(o)}_{i+n-1}$, and computing

$$h^{(LSE)} = (X^T X)^{-1} X^T v = X^+ v, \tag{4}$$

making use of the pseudo-inverse $X^+ = (X^T X)^{-1} X^T$. The estimates of the velocity $\hat{v}^{(o)}[k]$ needed for the LSE problem can be obtained from the sampled data by a suitable acausal filter, e.g. the popular Savitzky-Golay (SG) filter [3].

III. Empirical Verification

In this section, we present results from an empirical verification of the proposed adaptive velocity estimators on a difficult benchmark control problem: the stabilization of a rotary pendulum. The rotary pendulum, also known as the Furuta pendulum (FP) after the name of its inventor, has been a popular benchmark control problem used to investigate various control schemes [4]. It consists of two links, an arm and a pendulum, where the arm is actuated by means of applied torque τ around the vertical axis Z, and the pendulum rotates freely in a plane perpendicular to the arm, without any actuation. The state of the FP $x = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T$ is described by the arm and pendulum angles θ_1 and θ_2 and their angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$.

We are investigating the performance of a standard set-point stabilization controller, where the set-point is defined as the home position of the arm ($\theta_1 = 0$) and the upper, unstable equilibrium position of the pendulum ($\theta_2 = \pi$), under quantization noise for the second joint (θ_2). (That is, the desired set-point for regulation is $x_d = [\theta_{1,d}, \theta_{2,d}, \dot{\theta}_{1,d}, \dot{\theta}_{2,d}] = [0, \pi, 0, 0]^T$.) Although the proposed velocity estimation scheme is generally meant to be used for learning control applications, where a system model and/or controller will be learned from data, in this study, we tested its performance in conjunction with a manually designed stabilizing controller, in order to isolate and analyze only the effects of the velocity estimator on the stabilization performance. Following the approach in the original FP publication [4] as closely as possible, we designed an LQR FSF controller of the form $\tau[k] = -Kx[k]$, where the gains K were determined with cost matrices $Q = diag[1, 1, 1, 1]$ and $r = 1$ for the state variables and the control effort, respectively. In contrast to [4], where a physical model of the FP was derived and linearized analytically in order to design the stabilizing LQR controller, we created an FP model in the MuJoCo physics engine [5] and linearized it numerically around the set-point by means of finite differencing. Empirical verification of the velocity estimation algorithms was performed in MuJoCo using the full FP model.

The FP stabilization problem illustrates well the effect that encoder quantization noise has on the computed control signal. For a control rate of $F = 500Hz$, the computed LQR gains for the discrete-time controller were $K = [-0.67, 26.86, -1.14, 4.08]$. The negative gains for θ_1 and $\dot{\theta}_1$ reflect the need to first tilt the pendulum in the right direction by moving the arm *away* from its set-point before closing the error on the arm, if starting from a balanced position for the pendulum ($\theta_2[0] = \pi$) and at rest ($\dot{\theta}_1[0] = \dot{\theta}_2[0] = 0$), but with some error on the arm's angle ($\theta_1[0] \neq 0$). (The LQR controller typically performs very well in closing the error and bringing the FP to its target position even from significantly different starting arm angles $\theta_1[0]$, because the linearization of the FP remains valid, as long as the pendulum remains mostly upright during movement.) However, the opposite signs of the control gains also mean that often the resulting control signal, the torque τ applied to the arm, is the result of opposing feedback contributions computed through relatively large gains, even if the final value of the torque is not so high.

In particular, the gain $k_{\dot{\theta}_2}$ on the pendulum's velocity error is fairly large, $4.08\ \mathrm{Nm \cdot s/rad}$ in this case. When using simple consecutive encoder angle differences as velocity estimators, this relatively large gain ends up multiplying the noise $\Delta\epsilon[k] = \epsilon[k] - \epsilon[k-1]$ in these differences, as discussed in Section I. This noise $\Delta\epsilon[k]$ is a random variable obtained as a difference of two i.i.d. uniform random variables defined over the interval $[-\Delta/2, \Delta/2]$, where $\Delta = 2\pi/2^M$ rad is the width of a single sector of the angle encoder of resolution M bits. The probability distribution of $\Delta\epsilon[k]$ can be shown to be symmetric triangular, defined over the interval $[-\Delta, \Delta]$ and peaking at zero. Its mean is zero, but its magnitude, if characterized by its deviation from the mean, is by definition equal to the standard deviation of the triangular distribution, which is $\sigma = \Delta/\sqrt{6}$. This allows us to compute the expected magnitude of the disturbance that the quantization error in encoder angles will contribute to the control signal if used directly for velocity estimation, as $w = k_{\dot{\theta}_2} F\sigma = k_{\dot{\theta}_2} F\pi/(2^{M-1}\sqrt{6})$. If we employ a 10-bit encoder for the pendulum's angle, we obtain $w = 5.11$ Nm, which is more than four times larger than the maximal torque (around 1.2 Nm) needed to stabilize the pendulum even from arm angles that are 90 degrees away from the set-point ($\theta_1[0] = \pi/2$), if no disturbance was present. This demonstrates the challenges of using velocity estimates based on encoder measurements with large quantization errors in full-state feedback controllers on difficult control tasks, such as balancing an FP around its unstable equilibrium, and the importance of suppressing the effects of these quantization errors by means of more accurate velocity estimators.

IV. Results

In order to collect data for learning the FIR of the velocity estimator, the FP model was simulated in MuJoCo, starting from the lower stable equilibrium $x[0] = [0, 0, 0, 0]$ and applying random torques $\tau[k] \sim N(0, 8)\,\mathrm{Nm}$ and limited to 20 Nm in magnitude for a duration of 4 s and at a sampling rate of 500 Hz, resulting in $N = 2{,}001$ samples. The measured pendulum angle θ_2 was quantized at 1,024 levels, corresponding to a $M = 10$-bit absolute encoder.

A. Evaluation of the LSE Velocity Estimator's Accuracy

Following standard ML practice, the data set was split into two halves, the first for training and the second for testing. The angles $[k]$, $k = 0, \ldots, (N+1)/2$ in the training data set were filtered by an SG filter with window of size 15 and order 3 to remove as much quantization noise as possible, using SciPy's implementation of the SG filter [6]. This implementation also conveniently computes directly the derivative of the filtered time series, which becomes the sequence of best acausal, offline estimates $\hat{v}^{(o)}[k]$, $k = 1, \ldots, (N+1)/2$ to be used in (4) to minimize the squared error defined in (3).

Position differences $\Delta[k]$, $k = 1, \ldots, (N+1)/2$ for the training data set were also computed as inputs to the LSE estimator filter. Because the optimal order of the LSE estimator is not known, estimation was performed for a range of orders $n = 1, \ldots, 10$. Root mean squared (RMSE) velocity estimation errors for various estimators are shown in Fig. 1 for the training (top) and testing (bottom) sets, for one representative random generator seed. For the sake of fair comparison between filters of different orders, all filters' RMSE was computed over the same range of samples $\Delta[k]$, $k = 10, \ldots, (N+1)/2$.

The fitting error for each filter order, defined with respect to the best offline velocity estimates $\hat{v}^{(o)}[k]$ that it was computed from, is shown in Fig. 1 (top) under the label "LSE SG". This fitting error on the training set is monotonically non-increasing with the filter order, as higher-order filters have more parameters (filter coefficients), so their fitting error is necessarily lower or equal to that of lower-level filters. This property is not always true for the RMSE computed with respect to the true velocities recorded from the MuJoCo simulator, shown under the label "LSE"; the lowest RMSE there is achieved at $n = 2$. However, the RMSE of LSE(n) estimators for $n > 2$ remains largely comparable to that of LSE(2).

This is not at all the case for the simple differencing filters, labelled "SDE", that are also included in Fig. 1 (top), for the sake of comparison. Their FIR is equal to $h = [1/n, 1/n, \ldots, 1/n]$, for the n-th order SDE(n) filter. The SDE filters (not to be confused with the BDE filters, as defined in [1]) are equivalent to using the single back difference for velocity estimation as if the sampling rate were reduced n times, thus ignoring every $n-1$ out of n samples in the data set. After a significant drop for $n = 2$, their RMSE rises sharply for higher orders, reflecting the lag between the estimate of what is in fact the average velocity computed by

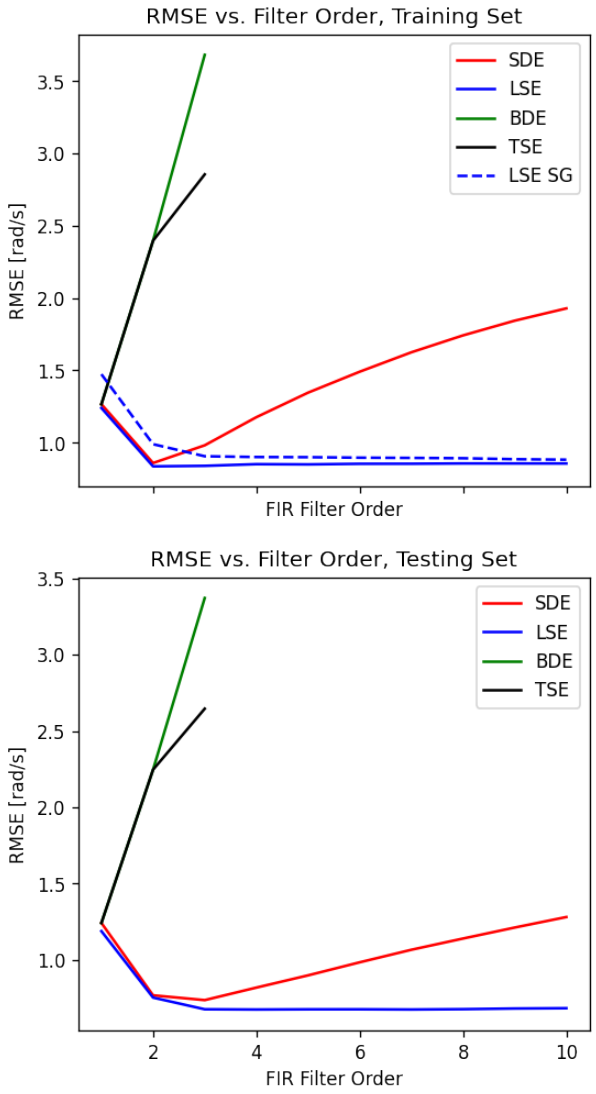

Fig. 1: RMSE of filters of increasing order on the training (top) and testing (bottom) data sets. The solid lines show RMSE computed with respect to the true velocities provided by the simulator, whereas the dashed line is computed with respect to the SG-filtered training data set, to show the goodness-of-fit achieved by the LSE routine. The SDE filters's RMSE increases sharply after its lowest value, whereas the LSE filters show little, if any, over-fitting.

finite differencing over increasingly longer time intervals, and the momentary velocity that is the correct target for estimation.

In contrast, the RMSE curve for the LSE estimator on the training set remains largely flat with the increasing filter order. Why this is the case can be understood by looking at the filter coefficients of the LSE estimators of different orders. These coefficients are shown in Fig.2, for $n = 1, \ldots, 6$. It can be observed that, whereas LSE filters of order 1 and 2 have substantially unique coefficients, all filters of order $n \geq 3$ are very similar, and have only increasingly minor differences in the longer lags, which are very small in magnitude, anyway. This suggests that the LSE estimation procedure has essentially figured out that the most recent three or so position differences $\Delta x[k-i]$, $i = 0, 1, 2$ are useful for predicting the momentary velocity, but position

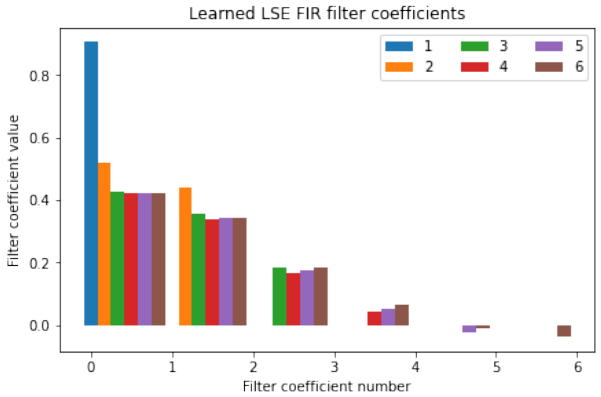

Fig. 2: Coefficients (FIR components) of the LSE estimators of orders $n = 1, \ldots, 6$ estimated from the training data set.

differences earlier than that are more harmful than useful and should be ignored during prediction.

Another way to understand the difference between the SDE and LSE estimators is to make use of the probabilistic analysis from the previous section. Recall that the standard deviation σ of the differences $\Delta\epsilon[k]$ in quantization noise terms of two consecutive samples is given by $\sigma = \Delta/\sqrt{6}$. The SDE(1) estimator uses the most recent angle difference $\Delta[k]$ as a velocity estimate after dividing it by the sampling period $T[k]$, or equivalently by multiplying it by the sampling rate F. As a result, the quantization noise $\Delta\epsilon[k]$ is propagated into the velocity estimate $\hat{v}^{(SDE(1))}$, contributing to its RMSE an error component equal to $F\Delta/\sqrt{6} = F\pi/(2^{M-1}\sqrt{6})$, which for the test system is equal to 1.25 rad/s. Fig. 1 shows that this is indeed very close to the sample RMSE of SDE(1) on both the training and testing data sets. The second component of the RMSE of SDE(1) is due to the time lag between the true momentary velocity at time $t[k]$ and the average velocity over the time interval $[t[k-1], t[k]]$ that the estimator is in fact computing. For SDE(1), this second component of the error appears to be negligible at this sampling rate, so the RMSE of SDE(1) is almost entirely due to quantization error.

SDE(2) halves the component of the error due to quantization, using an effective sampling rate of $F/2$, but incurs non-negligible error due to lag, which is now between the true momentary velocity at time $t[k]$ and its average over the twice longer time interval $[t[k-2], t[k]]$. For this reason, the RMSE of SDE(2) is lower than that of SDE(1), but not as low as half of it. With increasing filter order n, the component of SDE(n)'s RMSE due to lag starts to exceed the benefit of suppression of the quantization noise, and the overall RMSE goes up. In contrast, the LSE estimators learn the optimal weights (gradually decreasing with the filter coefficient's lag) that will not allow the lag error to exceed the error reduction from noise suppression, thus always resulting in better estimators with higher-order filters.

This analysis is also supported by the verification of the LSE filters' accuracy on the independent testing set that has not been seen during training, shown in Fig. 1 (bottom). In this graph, the filter LSE(n) of particular order n estimated on the training set was applied to the data in the testing set, and the resulting RMSE plotted against the filter order. The RMSE for the LSE filters is also largely flat for $n \geq 3$, increasing only very slightly, demonstrating that the proposed estimation method does not overfit the data. This claim can be quantified by computing the relative suboptimality (excessive RMSE) on the testing set of the LSE filter with the best fitting error on the training set (which will always be the highest-order filter, LSE(10) for this experiment), with respect to the LSE filter that happens to be optimal on the testing set, which is LSE(4) in this case. For this data set, the suboptimality of LSE(10) with respect to LSE(4) is only in the amount of 1.27% of the latter's RMSE, whereas its RMSE is 44.97% lower than that of the SDE(1) filter that is commonly used for velocity estimation. This shows that the differences between LSE filters of orders past $n = 3$ are very minor in comparison to the large improvement in accuracy they exhibit with respect to the most commonly used estimator in practice, SDE(1).

The accuracy of the BDE and TSE estimators of first, second, and third order are shown in Fig. 1, too, for the sake of comparison. Somewhat unexpectedly, their performance is much worse than that of the LSE filters. A possible explanation is that the learned coefficients of the LSE filters differ substantially in nature from the theoretically derived BDE and TSE ones. Even though the LSE coefficients vary somewhat depending on the particular random process realization from which the training data was collected, they are consistently within the interval $[0, 1]$, with negative coefficients of negligible magnitude only in the higher-order lags. This is in stark contrast with the FIR of BDE and TSE filters, where for $n \geq 2$, $h_0 > 1$ and $h_1 < 0$; for example $h = [1.5, -0.5]$ for BDE(2) (or, equivalently, TSE(2)). Under the assumption of i.i.d. quantization noise, each position difference $\Delta x[k]$ contributes to the total RMSE of the estimated velocity proportionally to the *absolute value* of its respective coefficient, so even if all coefficients in the FIR add up to a value close to one, it is the sum of their absolute values that determines the RMSE of the velocity estimate. As this sum grows with the filter order of the BDE and TSE filters, so does their RMSE.

This significant difference between the FIRs of BDE/TSE and LSE estimators also demonstrates that the proposed learning method does not merely fine tune estimators already known from the research literature, but learns substantially different estimators, tailored to the system whose velocity is being estimated.

B. Evaluation of the LSE Velocity Estimator's Performance in Control

Although the preceding results suggest that the LSE estimator is significantly more accurate than baseline estimators in predicting (estimating) momentary velocity, the ultimate test of its usefulness in control applications is whether its use would result in improved control performance. As noted, different velocity estimators based on causal FIR filters trade off quantization noise suppression for time lag, and their

RMSE on test data does reflect both components. However, the time lag might additionally impact the performance of the feedback controller, and this impact can be expected to be more significant for higher-order estimators.

For this reason, we compared the performance of several of the learned LSE estimators vs. that of baseline estimators on the difficult control problem we described above, FP balancing, that requires accurate velocity estimation. An (expected) optimal order of the LSE estimator can be determined from the RMSE curves above by means of cross-validation, as the LSE filter learned from the training data set that performs best on the testing data set (LSE(4) here).

The controller we tested was the aforementioned LQR controller computed for costs $Q = diag[1, 1, 1, 1]$ and $r = 1$ and resulting feedback gains $K = [-0.67, 26.86, -1.14, 4.08]$, on the task of stabilizing the FP around its upper, unstable equilibrium, for a duration of 2 s ($N_c = 1,001$ control steps). The controller was tested from two different initial states. The first was the unstable equilibrium itself, $x[0] = x_d$. Even though the system starts in the goal state, the pendulum encoder immediately introduces a finite quantization error in the pendulum's angle, prompting the controller to act, applying a torque that brings the pendulum out of equilibrium, and from then on, the controller is actively balancing the inverted pendulum by applying torque to the arm to counteract the disturbance.

The second initial state was also with the pendulum at its unstable upper equilibrium and at rest, but the arm starting at a right angle to its goal state: $x[0] = [\pi/2, \pi, 0, 0]^T$. This is a much more difficult control problem, because bringing the FP from its initial position to the goal position requires maneuvering the arm carefully, first away from the goal state, and then back towards it, while keeping the pendulum tilted the right way, and finally straightening and balancing the pendulum upright.

The mean squared regulation errors for the four state variables, as well as their totals, are shown in Tables I and II for the stabilization and transportation control tasks (i.e., with different initial states) under several velocity estimators. An evaluation for the BDE(2) (equivalently, TSE(2)) estimator with FIR $h = [1.5, -0.5]$ ([1]) is included, too.

The results show that all velocity estimators, BDE and LSE, in conjunction with the LQR controller, were able to bring the FP to the desired goal state and balance it there successfully, without ever dropping it. Their regulation performance, though, varied widely. For both tasks, the best overall performance was achieved with the learned LSE estimator of second order, LSE(2). Its mean squared cost was 28% and 14% lower than that of the second-best estimator, BDE(1), on the two tasks, respectively. It can also be seen that the reduction in regulation error was achieved almost entirely in the velocity terms.

These results suggest that the proposed LSE estimators can improve the quality of regulation with respect to well-known estimators with fixed impulse responses, such as the BDE and TSE estimators. However, the optimal order of the LSE estimator on the control tasks is not necessarily the one with the lowest estimation error; this could be due to the lag introduced by higher-order estimators into the control loop. Furthermore, the second-order LSE estimator had much better performance than the BDE/TSE estimator of the same order, suggesting that the proposed adaptive data-driven procedure can produce impulse response coefficients that are much more appropriate and effective for control purposes than those derived theoretically.

Estimator	θ_1	θ_2	$\dot{\theta}_1$	$\dot{\theta}_2$	Total
BDE(1)	**0.00008**	**0.000006**	10.22	1.97	12.19
BDE(2)	0.01031	0.000337	40.81	7.81	48.64
LSE(2)	0.00022	0.000014	**7.35**	**1.42**	**8.78**
LSE(3)	0.00032	0.000041	21.57	4.16	25.72
LSE(4)	0.00055	0.000069	32.56	6.28	38.84

TABLE I: Mean squared error with several velocity estimators on the FP *stabilization* task, tabulated by state variable and in total. Best (lowest) MSE across estimators is displayed in bold.

Estimator	θ_1	θ_2	$\dot{\theta}_1$	$\dot{\theta}_2$	Total
BDE(1)	0.580	0.000107	10.59	1.99	13.17
BDE(2)	0.542	0.000921	41.05	7.68	49.28
LSE(2)	0.568	0.000111	**9.02**	**1.69**	**11.28**
LSE(3)	**0.540**	**0.000106**	18.47	3.51	22.54
LSE(4)	0.582	0.000144	34.14	6.53	41.25

TABLE II: Mean squared error with several velocity estimators on the FP *transportation* task, tabulated by state variable and in total.

V. Conclusion and Future Work

In this paper, we presented a method for learning velocity estimators in the form of an FIR filter from data collected from a system equipped with quantizing position encoders that will be controlled by means of a full state controller making use of velocity estimates. The method performs least-squares estimation of the impulse response coefficients, using off-line estimates of the momentary velocities obtained by smoothing or another suitable signal denoising and reconstruction method applied on a training data set. Empirical verification in simulation demonstrates that the LSE velocity estimators exhibit up to 5 times lower velocity prediction error than simple differencing filters of the same order, and are not prone to overfitting. In future work, we plan to investigate better off-line signal reconstruction methods that might improve the accuracy of the learned estimators further.

References

[1] R. H. Brown, S. C. Schneider, and M. G. Mulligan, "Analysis of algorithms for velocity estimation from discrete position versus time data," *IEEE Transactions on industrial electronics*, vol. 39, no. 1, pp. 11–19, 1992.

[2] R. J. E. Merry, M. J. G. Van de Molengraft, and M. Steinbuch, "Velocity and acceleration estimation for optical incremental encoders," *Mechatronics*, vol. 20, no. 1, pp. 20–26, 2010.

[3] R. W. Schafer, "What is a Savitzky-Golay filter?" *IEEE Signal processing magazine*, vol. 28, no. 4, pp. 111–117, 2011.

[4] K. Furuta, M. Yamakita, S. Kobayashi, and M. Nishimura, "A new inverted pendulum apparatus for education," in *Advances in Control Education 1991*. Elsevier, 1992, pp. 133–138.

[5] E. Todorov, "Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6054–6061.

[6] Eric Jones, Travis Oliphant, Pearu Peterson, and others, "SciPy: Open Source Scientific Tools for Python," 2001. [Online]. Available: http://www.scipy.org/