applied sciences

MDPI

Article

Comparing Methods of DC Motor Control for UUVs

Rohan Shah [1] and Timothy Sands [2,*]

[1] Systems Engineering, Cornell University, Ithaca, NY 14853, USA; rns85@cornell.edu
[2] Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA
* Correspondence: tas297@cornell.edu

Featured Application: Underwater vehicle control surfaces motor control.

Abstract: Adaptive and learning methods are proposed and compared to control DC motors actuating control surfaces of unmanned underwater vehicles. One type of adaption method referred to as model-following is based on algebraic design, and it is analyzed in conjunction with parameter estimation methods such as recursive least squares, extended least squares, and batch least squares. Another approach referred to as deterministic artificial intelligence uses the process dynamics defined by physics to control output to track a necessarily specified autonomous trajectory (sinusoidal versions implemented here). In addition, one instantiation of deterministic artificial intelligence uses 2-norm optimal feedback learning of parameters to modify the control signal, while another instantiation is presented with proportional plus derivative adaption. Model-following and deterministic artificial intelligence are simulated, and respective performance metrics for transient response and input tracking are evaluated and compared. Deterministic artificial intelligence outperformed the model-following approach in minimal peak transient value by a percent range of approximately 2–70%, but model-following achieved at least 29% less error in input tracking than deterministic artificial intelligence. This result is surprising and not in accordance with the recently published literature, and the explanation of the difference is theorized to be efficacy with discretized implementations.

Keywords: path planning; dynamics; modeling; mechanics; adaption; deterministic artificial intelligence; parameter estimation; least squares; proportional-derivative; model-following; motor control

Citation: Shah, R.; Sands, T. Comparing Methods of DC Motor Control for UUVs. *Appl. Sci.* **2021**, *11*, 4972. https://doi.org/10.3390/app11114972

Academic Editor: Juan-Carlos Cano

Received: 30 April 2021
Accepted: 27 May 2021
Published: 28 May 2021

1. Introduction

Adoption of computers and automation revolutionized manufacturing and optimization of the computerization is deemed to be continuing the revolution. Like the manufacture of DC motors, motor operations are also amidst revolutionary change. DC motors are commonplace mechanisms to actuate control surfaces and rotate propellors of unmanned underwater vehicles, as depicted in Figure 1. Example motors are depicted in Figure 2, where current is supplied to wires at one end of the motor and rotation is provided by the motor at the other end (as depicted). Control of the motor is a well-studied topic in the literature [1–4], culminating in a very recent publication of deterministic artificial intelligence used for motor control [5,6]. Motor control using neural networks was presented in [1–3], while indirect self-tuners were presented in [4], which were the basis for comparison of the recently presented method of deterministic artificial intelligence [5], where the prequel comparison to self-tuners is presented in [6] applied to DC motors for unmanned underwater vehicles.

This manuscript firstly seeks to validate the results of the recently proposed application of deterministic artificial intelligence [6] to a disparate motor to reveal the general efficacy of the methodology. Secondly, this manuscript evaluates the efficacy of the proposed approach in direct comparison to state-of-the-art methods: model-following methods with various methods of parameter estimation including batch least squares, recursive least squares, and extended least squares.

Figure 1. Control surfaces and propellor on unmanned underwater vehicles [7] utilizing typically available DC motors [8,9].

(**a**) (**b**) (**c**)

Figure 2. (**a**) Maxon high-torque DC brushless motors for UUVs [8]. (**b**) Underwater thruster propeller motor [9]. (**c**) ECI-40 Maxon underwater drive motor and gear head.

1.1. Literature Review

At the ancient age of 18 years old, Gauss developed the method of least squares in 1795, although Legendre is often credited with its development in 1805 [10] and subsequent (significant) expansion as stochastic methods [11,12] in the following years [13–16]. Gauss used the method to determine the motion of heavenly bodies about the sun in conic sections [10], subsequently focusing on errors of estimation and prediction fitting of linear relationships (mathematical models). Gauss's thinking helped establish one of the first international journals in the field of astronomy [13], founded in 1821 by the German astronomer Heinrich Christian Schumacher. It claims to be the oldest astronomical journal in the world that is still being published [14].

Legendre is credited with developing the least squares method to use only measurements to learn about natural processes.

> *To determine the orbit of a heavenly body, without any hypothetical assumption, from observations not embracing a great period of time, and not allowing the selection with a view to the application of special methods.* [15,16]

The least squares method forms the basis of estimation for several adaptive modeling methods described by Åström and Wittenmark in [17–19] using the recursive forms compared in [20,21], while the batch form is emphasized by Slotine in [22], which was improved as an adaptive method by Fossen in [23] and subsequently by Sands in [24]. The feedback measurements emphasized by Gauss are parameterized in a Diophantine equation [25], also referred to in the literature as a Bezout identity [26], or, alternatively, as an Aryabhatta equation [27] discussed relative to each other in [27]. Without overtly saying so, system identification is described [28] where the time-varying terms in a nonlinear adaptive controller are the identified system parameters.

For a system of unknown parameters, several methods of adaption can be implemented to control the system and achieve a desired response. Often, a desired response is

rapid tracking of an input signal by an output signal [29]. Most natural processes do not cause a change in input to be matched by its resulting output, so certain adaptive methods can allow for control and tracking using physics-based methods of Lorenz [30], which were experimentally validated in [31] as applied to space systems.

The main difference between nonlinear adaptive control as presented by Slotine [29] and the physics-based method offered by Lorenz [30] is the time-invariant nature of the mathematical expression of modeling by physics, where Slotine adapts a form of the mathematical expression to counter deleterious error effects. The general spirit of both is embodied in the feedforward portion of the recently proposed deterministic artificial intelligence (wh's process flow is illustrated in Figure 3 and who's the self-awareness statements are depicted in Figure 4). Parameter adaption (per Slotine) is one option, while optimal learning (per Smeresky in [32]) is another option which utilizes variations of the aforementioned least squares methods of Gauss.

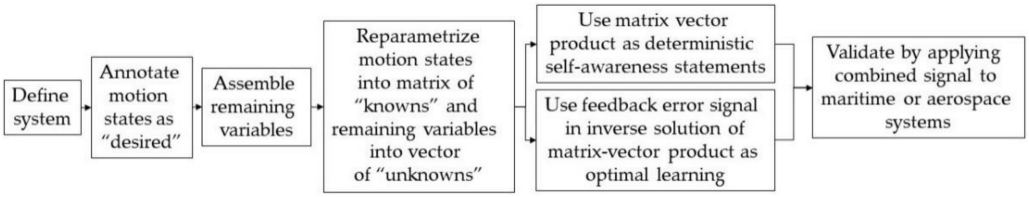

Figure 3. Topology of deterministic artificial intelligence whose depiction applied to UUV system whose actuators are powered by DC motors is in Figure 4.

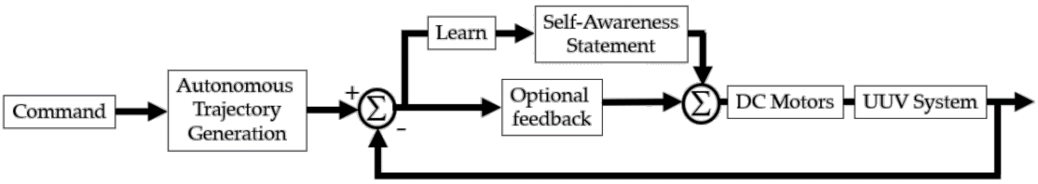

Figure 4. Topology of deterministic artificial intelligence applied to UUV system whose actuators are powered by DC motors.

Two main avenues studied to achieve input tracking are through controller design for self-tuning regulators or through learning parameters and modifying an input signal without changing the process dynamics of the system [33,34]. Self-tuning regulators allow for process control through algebraic design and parameter estimation, so an accurate controller is developed as process parameters are estimated over time. These methods form the benchmark for comparison to recently published adaption and learning methods.

Learning methods like deterministic artificial intelligence [5] whose topology is depicted in Figure 4, involve using the process dynamics (dictated by mathematical models of physics) as a form of control [6], and evaluation is done by analyzing input tracking performance metrics. It noteworthy to indicate self-awareness statements in deterministic artificial intelligence only function when supplied an analytic desired trajectory, where error calculation necessitates state observers [35] which enable both adaption or 2-norm optimal learning according to recently published results by Smeresky [32]. Foreshadowing this manuscript's Materials and Methods section, where the method of deterministic artificial intelligence is formally presented, the brevity of its elaboration (merely two necessary equations with explanatory verbiage) is testament to its strength (simplicity) compared to the other methods compared.

The challenges, motivation, and literature gap this work aims to address include application of previously published methods to a new motor model.

1.1.1. Challenges

Step functions are challenging, discontinuous demanded trajectories and square waves compound the challenges by repeatedly reversing direction with discontinuous steps. One

key unsolved challenge remains tracking such demanded trajectories without dramatic overshoot and settling.

1.1.2. Motivation

Previously published studies indicated efficacy of model-following techniques and also deterministic artificial intelligence applied to a certain kind of DC motor. This work is motivated to investigate efficacy of various parallel instantiations of model-following and deterministic artificial intelligence on a disparate motor to ascertain the general applicability of the methods, particularly in discrete-time (unlike the sequel [6]).

1.1.3. Literature Gap

The study in [6] represents a most recently published addition to the literature and this manuscript extends that most recent publication to a disparate system. The study in [6] uses self-tuning regulators with a continuous plant while model-following designs are utilized here with a discretized plant (of a different motor than used in [6]).

1.2. Developments Presented

Validation is attempted of recently proposed methods on a disparate motor: this manuscript applies methods to a different motor model than recently published and illustrates comparison of transient response and input tracking using disparate adaptive techniques as the benchmark for comparison. Deterministic artificial intelligence outperformed the model-following approach in minimal peak transient value by a percent range of approximately 2–70%, but model-following achieved at least 29% less error in input tracking than deterministic artificial intelligence This relatively poorer performance (compared to the recently published results) in tracking error led to impromptu studies of possible explanations, and those studies led to the second development presented in this manuscript. Illumination of key, unpublished facet: in this study, deterministic artificial intelligence illustrated relatively greater performance degradation using discretized implementations revealing a unique weakness, namely relatively higher reliance on computational capacity that may preclude application of the technique on unmanned underwater systems like spacecraft, aircraft, and other such systems where constraints on computation capacity are driven by trade-off decisions with weight and cost. A short summary of novel contributions follows:

1. Validation of recently published proposal to use deterministic artificial intelligence for motor control by application to a disparate motor.
2. Comparison of four disparate methods (including model-following) paralleling comparisons just published.
3. Illumination of a limitations due to discretization with deterministic artificial intelligence.

Section 2 of this manuscript introduces the mathematical motor model and introduces model-following control design presented by Åström and Wittenmark in [19] and compared in [21]. The newly proposed method of deterministic artificial intelligence [6] is introduced next. The methods are compared in Section 3 firstly emphasizing estimation performance followed by adaptive model-following and then deterministic artificial intelligence. Lastly, Section 4 discusses the results and how they can be interpreted from the perspective of previous studies and of the working hypotheses. The findings and their implications are discussed in the broadest context possible and future research directions are also be highlighted.

2. Materials and Methods

This section introduces the mathematical motor model and introduces model-following control design. The newly proposed method of deterministic artificial intelligence is introduced next, where methods and protocols are described in detail while well-established methods can be briefly described and appropriately cited.

2.1. Truth Model for Motor Dynamics

The process truth model for natural dynamics of a DC motor as described by the sequel [6] in Equation (1) is shown in frequency domain form using the z-transform for discrete-time signals, while the motor investigate here is included in Equation (2). Notice both the disparate Equations (1) and (2) and the final discretization utilized here (2).

$$G(s) = \frac{B(s)}{A(s)} = \frac{1}{s(s+1)} \rightarrow G(z) = \frac{0.09842z + 0.09842}{z^2 - 1.607z + 0.6065} \tag{1}$$

$$\frac{B(z)}{A(z)} = \frac{b_0 z + b_1}{z^2 - a_1 z - a_2} = \frac{0.1065z + 0.0902}{z^2 - 1.9z + 0.88} \tag{2}$$

The process equation, known as a transfer function, has a zero at $z = -0.8469$ and poles at $z = 1.1,\ 0.80$. The transfer function has an unstable pole, and the approach with model-following will effectively relocate these poles to stable locations at $z = 0.2 \pm 0.2j$. The deterministic artificial intelligence modeling approach does not attempt to achieve pole relocation, but rather achieves an autonomously determined trajectory using proportional plus derivative (PD) feedback adaption of unknown parameters to follow the target path. The truth model also incorporates correlated Gaussian noise with distribution $N\left(0, \frac{1}{625}\right)$ added as two delayed noise terms. Simulations were performed for both modeling approaches using scripts developed in MATLAB, and the code for each implementation can be found within Appendix A. Identical square wave inputs signals were fed to model-following control designs as well as deterministic artificial intelligence, while the latter method includes an autonomous path planning algorithm to create sinusoidal trajectories that commence at the starting discontinuity of the square wave and end at the peak of each square wave discontinuity.

2.2. Model-Following Motor Control Design

The designed control system implemented a dynamic feedforward for input u_c and negative dynamic feedback for output y that is summed to produce the process input u, as shown in Figure 3.3 in [19]. The model-following topology can be directly compared with the deterministic artificial intelligence topology depicted by Figure 2 in [5]. The system response can be described by Equation (3), with $U(z)$ equal to the z-transform of the control signal, and $Y(z)$ equal to the z-transform of the control output.

$$Y(z) = \frac{B}{A}\left(\frac{T}{R}U(z) - \frac{S}{R}Y(z)\right), \quad \frac{Y(z)}{U(z)} = \frac{BT}{AR + BS} \tag{3}$$

As the designed topology allows for certain cancellations to occur within Equation (3), the numerator and denominator can be represented as a product of factors as well in Equation (3), where B^+ represents the cancelled zeros, B^- represents uncancelled zeros, B'_m represents a scalar multiple to the system, and A_0 represents the pole-zero cancellations used to generate the desired system transfer function.

$$\frac{Y(z)}{U(z)} = \frac{B^+B^-A_0B'_m}{A_0A_mB^+} = \frac{BT}{AR + BS} = \frac{B_m}{A_m} \tag{4}$$

Since the demonstrated control design will involve no process zero cancellations, B^+ is set to 1. According to causality conditions, the polynomials R, S, and T are of first order. By manipulation of Equation (4), the coefficients of polynomials R′, S, and T can be defined in terms of estimated and desired process parameters in Equations (5)–(7), which are adapted from [19].

$$r_1 = \frac{b_1}{b_0} + \frac{\left(b_1^2 - a_{m_1}b_0b_1 + a_{m_2}b_0^2\right)\left(-b_1 + a_0b_0\right)}{b_0\left(b_1^2 - a_1b_0b_1 + a_2b_0^2\right)} \tag{5}$$

$$s_0 = \frac{b_1\left(a_0a_{m_1} - a_2 - a_{m_1}a_1 + a_1^2 + a_{m_2} - a_1a_0\right)}{b_1^2 - a_1b_0b_1 + a_2b_0^2} + \frac{b_0(a_{m_1}a_2 - a_1a_2 - a_0a_{m_2} + a_0a_2)}{b_1^2 - a_1b_0b_1 + a_2b_0^2} \tag{6}$$

$$s_1 = \frac{b_1(a_1a_2 - a_{m_1}a_2 + a_0a_{m_2} - a_0a_2)}{b_1^2 - a_1b_0b_1 + a_2b_0^2} + \frac{b_0\left(a_2a_{m_2} - a_2^2 - a_0a_{m_2}a_1 + a_0a_2a_{m_1}\right)}{b_1^2 - a_1b_0b_1 + a_2b_0^2} \quad (7)$$

With the added feedforward and feedback signals, the control signal into the process can now be described by Equation (8) to Equation (9). The factor of β in Equation (9) will cause the system response to have unity gain, which is important for attaining zero asymptotic tracking error [19].

$$RU(z) = TU_c(z) + SY(z) \quad (8)$$

$$U(t) = \beta U_c(t) + \beta a_0 U_c(t-1) + s_0 Y(t) + s_1 Y(t-1) - r_1 U(t-1) \quad (9)$$

2.3. Deterministic Artificial Intelligence Methodology

Through modeling with deterministic artificial intelligence, changes in input do not result in a drastically forced change in output to match the input signal value. Instead, with change in state of the input, a trajectory (sinusoidal here) is calculated for the output to follow in progression towards the target state so there is no undefined position (analytically) for any timestep. Unlike model-following, dynamics of the process are *asserted* as the control mechanism in a feedforward fashion establishing self-awareness, and the control signal can be modulated through feedback parameters that are learned using a proportional-derivative feedback mechanism or 2-norm optimal methods. Equation (10) shows how feedback parameters can be calculated through batch least squares per Smeresky in [32], with ϕ_d representing desired trajectory states, $\hat{\theta}$ representing learned parameters to adjust control input u, and δu representing error in control input.

$$u = \phi_d\hat{\theta} = \phi_d\left(\phi_d^T\phi_d\right)^{-1}\phi_d^T\delta u \quad (10)$$

Proportional plus derivative parameter adjustment of $\hat{\theta}$ from [28] with proportional adaption gain of two and derivative adaption gain of six was used in the presented experiments in combination with assertion of self-awareness (original process dynamics described in Equation (11)), where Equation (10) describes optimal feedback adjustment described in [32], where no tuning is required for self-awareness and learning. Equation (11) embodies the static physics-based feedforward mentioned in the literature review of Section 1.1.

$$u \equiv \phi_d\theta \quad (11)$$

Equation (12) shows the sinusoid-based trajectory calculation for a change in input, where A is the target state, A_0 is the original state, ω is the frequency corresponding to maneuver speed, and ϕ is the phase offset of the sinusoid [36].

$$z = (A - A_0)[1 + sin(\omega t + \phi)] \quad (12)$$

When a change in state is detected, Equation (12) can be utilized to analytically determine a desired trajectory rather than relying on a model-following controller to stabilize the system. In the presented simulation, the sinusoidal frequency $\omega = 0.1\pi$ and phase $\phi = -\frac{\pi}{2}$. Using the specified parameters for trajectory calculation in response to changes in state with the square wave allowed for a smooth transition in state with a half-wave of a sinusoid instead of an abrupt change (e.g., associated with step functions) by magnitude $(A - A_0)$.

2.4. Analysis and Prediction

Comparison of Equation (1) to Equation (2), the two disparate motor models, indicates disparate results are possible compared to the just published literature [6]. Since the model-following methods drive performance to follow a desired model, the deterministic artificial intelligence method is predicted to produce smaller tracking errors, since the method does not impose a desired model, instead utilizes the model determined by mathematical physics. The prequel realized this prediction achieving 4.8% lower mean and 211% lower standard deviation of tracking errors as compared to the best modeling method investigated (indirect self-tuner without process zero cancellation and minimum phase plant). This

manuscript applies deterministic artificial intelligence in accordance with the prequel [6] but compares the performance to other state of the art (model-following) methods, applying both methodologies to a new motor model. Since indirect self-tuning regulators also utilize a preferred model to tune performance [37], the model-following results are predicted here to perform well, but with slightly inferior tracking errors.

3. Results

3.1. Plant Parameter Estimation for Model-Following Benchmark

Dynamic systems can often be modeled in terms of a finite set of parameters that can be methodically estimated through minimization of the summed squared difference between a fitted model and truth model. Batch least squares (LS), recursive least squares (RLS), and extended least squares (ELS) are examples of such estimation methods. RLS incorporates an operator for multiplication with previous parameter estimates to use error between true values and estimates as a technique to update the parameter vector to its most likely state for a specific timestep (determined by the specified discretization). Discretization timestep will prove to be a very important facet of implementation. As an extension of RLS, ELS is identical to RLS, where the system vector ϕ holds values of process input and output. Using ELS, values for noise, which may be correlated, are included in the vector ϕ. Such a structure is important when a process has naturally occurring noise, resulting in the need for an estimation of contribution to the system from noise to estimate process parameter values more accurately. The third technique tested, known as Batch LS, allows for a least squares estimation of process parameters based on a collected amount of observed input and output data, minimizing error for all observations rather than one at a time.

Based on the formulation of the RLS, ELS, and Batch LS parameter estimation techniques, a simulation was conducted to determine estimation performance of each approach. Plots in Figure 5 depict the convergence of each estimation method towards true values for two of four total unknown parameters over the 200 timestep simulation.

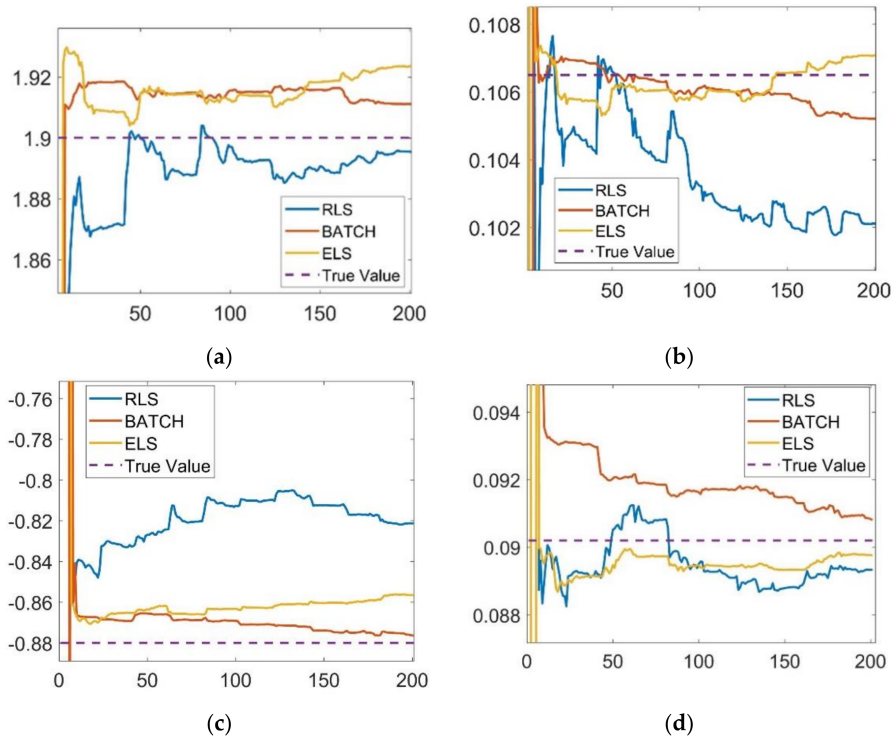

Figure 5. (**a**) Estimation of parameter a_1 on ordinate with time in seconds on the abscissa. (**b**) Estimation of parameter b_0 on ordinate with time in seconds on the abscissa. (**c**) Estimation of parameter a_2 on ordinate with time in seconds on the abscissa. (**d**) estimation of parameter b_1 on ordinate with time in seconds on the abscissa.

As shown in Figure 5 and Table 1, estimation using ELS performed the same or better than implementations of RLS and Batch LS. ELS had an error 9% higher than RLS for estimation of $\boldsymbol{a_1}$ but performed better in estimating the parameters of $\boldsymbol{a_2}$, $\boldsymbol{b_0}$, $\boldsymbol{b_1}$ with equal or lesser error averages and standard deviations than ELS and Batch LS. RLS performed similarly to ELS in estimating parameters $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$, and Batch LS exhibited high error in comparison to RLS and ELS.

Table 1. Mean/Standard Deviation of Estimation Technique Errors.

Estimated Parameter	True Value	RLS	ELS	Batch LS
a_1	1.9000	0.011/0.007	0.012/0.005	0.025/0.004
a_2	−0.8800	0.020/0.009	0.020/0.002	0.021/0.003
b_0	0.1065	0.002/0.001	0.0005/0.0002	0.001/0.001
b_1	0.0902	0.003/0.002	0.0004/0.0002	0.001/0.0007

3.2. Adaption Using Model-Following and Various Parameter Estimation (Benchmarks)

Model-following involves the design of a control topology that effectively converts an unstable system response to a stable one, which prevents divergence of the output and allows for inputs to be tracked asymptotically. In addition, model-following can incorporate parameter estimation techniques like RLS, ELS, and Batch LS to iteratively adapt the designed control topology to the estimated process characteristics, which is known as a self-tuning regulator.

Deterministic artificial intelligence (deterministic artificial intelligence) takes a different approach to input tracking, where an autonomous trajectory for output is defined for changes in input using a half-wave of a sinusoid. With the deterministic artificial intelligence modeling approach, natural system dynamics are asserted instead of modifying the existing process through control design as is done in model-following. In Figure 6a, a contrast between deterministic artificial intelligence and model-following performances exists in the transient responses with either approach. The transient with deterministic artificial intelligence has a comparably smaller peak, with a peak magnitude lower than that of RLS by 2.66%, ELS by 43.24%, and Batch LS by 67.18%. The discrete-time square wave depicted in Figure 6 was created with a sampling frequency of 1, displaying a slight slant visible in the plot at times of state change due to the short time axis length and nature of discrete (integer) sampling. The reference signal functionally acts as a square wave for discrete-time systems.

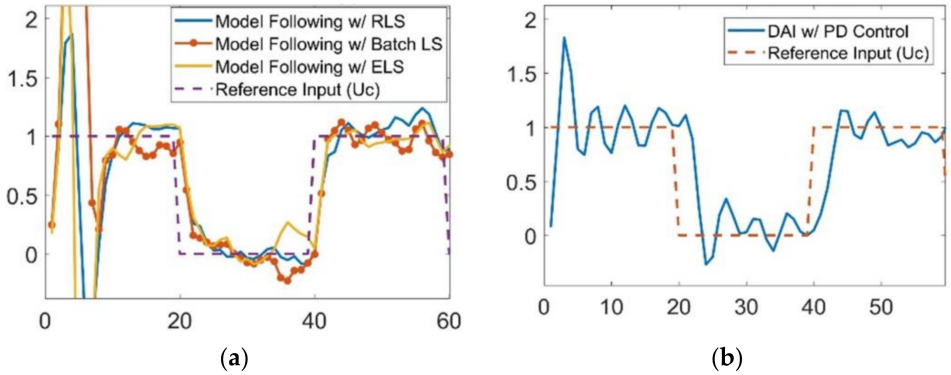

Figure 6. Results based on discretization of plant using difference equations in Equation (2) (**a**) Input tracking with model-following and parameter estimation on ordinate with time in seconds on the abscissa. (**b**) Input tracking with deterministic artificial intelligence and proportional plus derivative adaption of self-awareness statements on ordinate with time in seconds on the abscissa.

As shown in Table 2, output with the deterministic artificial intelligence modeling approach results in a significant average input tracking error at least 40.88% higher than

error resulting from implementations of model-following with different parameter estimation techniques. The high error is also reflected in Figure 6a, in which the output with deterministic artificial intelligence visibly fluctuates heavily after each change in input. Error distributions between each implementation of model-following are similar, with each average error being within a 9% margin of each other. It is evident, however, that there is a sharp difference in tracking accuracy between model-following and deterministic artificial intelligence. Seeking to elaborate the surprise difference in performance (especially relative to the superior performance of deterministic artificial intelligence achieved in [6], the results from that study are replicated in Figure 7 and Table 3. The key distinguishing feature is discretization where the present study implemented discretization while the study of [6] did not, and this difference revealed the surprising discovery constituting an unexpected novel contribution of this manuscript: *performance of deterministic artificial intelligence is severely degraded by discretization implementation compared to the prequel* [6] *using Equation (1) whose results are depicted in Figure 7.*

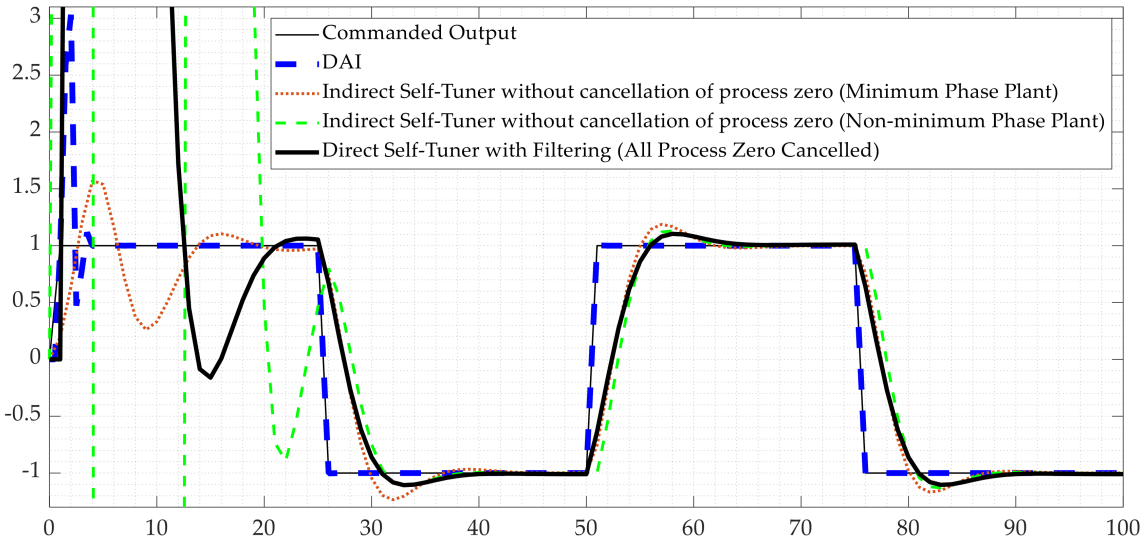

Figure 7. Results based on Equation (1) as opposed to Equation (2). Repeated from [6] comparing disparate modeling techniques (self-tuners without cancellation of process zeros) versus the identical deterministic artificial intelligence method used in this study. In this figure output is on ordinate with time in seconds on the abscissa. The reader is urged to compare the blue, thick dashed line in Figure 7 to the deterministic artificial intelligence results presented in Figure 6b. The difference is elaborated in Section 3.3.

Table 2. Mean/Standard Deviation of Input Tracking Errors Using Equation (2).

Model F. w/RLS	Model F. w/ELS	Model F. w/Batch LS	Deterministic Artificial Intelligence
0.149/0.247	0.147/0.224	0.159/0.245	0.224/0.288

Table 3. Mean/Standard Deviation of Input Tracking Errors Corresponding to Figure 7.

Direct STR	Indirect STR: No Zero Cancellation	Indirect STR: Zero Cancellation	Deterministic Artificial Intelligence
−0.35445/2.9984	23.2272/109.7158	0.023531/0.58929	−0.012239/0.1895

3.3. Analysis

The results presented in Section 3.2 highlight a contribution: deterministic artificial intelligence performance seems quite sensitive to discretization. Utilization of Equation (1) compared to Equation (2) clearly degrades performance of all methods used; but when compared to each other, the performance of deterministic artificial intelligence degrades

relatively more making the approach less desirable in instances where small discretization is not possible due to pressures on computer abilities (e.g., unmanned underwater vehicles, space systems, aircraft). Deterministic artificial intelligence uses optimal feedforward combined with either optimal learning in a 2-norm sense or alternatively nonlinear adaption. While model-following designs presented here were designed in discrete time [19].

4. Discussion

This research attempted to duplicate the just-published application of deterministic artificial intelligence to DC motors used on unmanned underwater vehicles. The just-published prequel makes comparisons to state-of-the-art methods in the category of self-tuning regulators, while this manuscript compares performance (of a disparate DC motor) to state-of-the-art methods in the category of model-following algorithms. Deterministic artificial intelligence outperformed the model-following approach in minimal peak transient value by a percent range of approximately 2–70%, but model-following achieved at least 29% less error in input tracking than deterministic artificial intelligence. This result is surprising and not in accordance with the recently published literature, and the explanation of the difference is shown to be efficacy with discretized implementations. In the prequel [6] utilizing similar estimation methods, deterministic artificial intelligence yielded 4.8% lower mean and 211% lower standard deviation of tracking errors as compared to the best modeling method investigated in that article: indirect self-tuner without process zero cancellation and minimum phase plant. The implication is the increased reliance by deterministic artificial intelligence on small nondiscretization size to replicated continuous systems.

Future Research

Clearly detailed investigation of the limits of permissible discretization using deterministic artificial intelligence is demanded by the results presented here. Furthermore, continued improvements in modeling and estimation seem logical. Comparison of the results here to model predictive control, as articulated by Marusak in [38,39] for prediction and by Nebeluk and Ławryńczuk for tuning in [40], seems to be very interesting for future research. In [41], Pappalardo proposes alternative uses of forward and reverse dynamics in the control, and comparison to the self-awareness feature of deterministic artificial intelligence seems to be a natural area for future improvements. Pappalardo also proposed developments to system identification [42] that could prove useful. Bruzzone recently introduced motor control based on fractional-calculus [43], and it seems that expression of self-awareness statements in deterministic artificial intelligence could be amplified.

Author Contributions: Conceptualization, R.S. and T.S.; methodology, R.S. and T.S.; software, R.S. and T.S.; validation, T.S.; formal analysis, R.S. and T.S.; investigation, R.S. and T.S.; resources, T.S; writing—original draft preparation, R.S.; writing—review and editing, R.S. and T.S.; supervision, T.S.; funding acquisition, T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data supporting reported results can be found by contacting the coordinating author at tas296@cornell.edu.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Algorithms (actual MATLAB computer code used) to simulate system responses for adaption methods. Adapted from example code provided through course material.

1. **Algorithm A1**: Recursive Least Squares
2. **Algorithm A2**: Auto-regressive moving average
3. **Algorithm A3**: Extended Least Squares
4. **Algorithm A4**: Deterministic Artificial Intelligence

Algorithm A1 Recursive Least Squares

```
clear all;clc;
%%%%%%%%%%%%%%%%RECURSIVE LEAST SQUARES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rand('seed',1);

B=[0 0.1065 0.0902];A=poly([1.1 0.8]);
a1=0;a2=0;b0=0.1;b1=0.2;
Am=poly([0.2+0.2j 0.2−0.2j]);Bm=[0 0.1065 0.0902];
am0=Am(1);am1=Am(2);am2=Am(3);a0=0;

Rmatrix=[];
factor = 25;
%create square wave for reference input
maxtime=200;
Uc = zeros(1,201);
for i=1:length(Uc)
    if (mod(floor(i/20),2) == 0)
        Uc(i) = 1;
    else
        Uc(i) = 0;
    end
end
%Uc = [ones(1,50) -ones(1,50) ones(1,50) -ones(1,51)];
traj_Uc = zeros(1,length(Uc));
check = 1;
run_next = 0;
for i=1:length(Uc)-1
    if (check)
        traj_Uc(i) = Uc(i);
        diff = Uc(i+1)-Uc(i);
        lasti = i;
         lastval = Uc(i);
    end
    if (diff ~= 0)
         check = 0;
        if (run_next)
            traj_Uc(i) = lastval + diff/2*(1+(sin(0.2*pi*(i-lasti)-pi/2)));
        end
        run_next = 1;
        if (traj_Uc(i) == Uc(i) && (i ~= lasti))
            disp('enter');
            check = 1;
            run_next = 0;
        end
    end
end
Uc = Uc(1:200);

n=4;lambda=1.0;
nzeros=5;time=zeros(1,nzeros);Y=zeros(1,nzeros);Ym=zeros(1,nzeros);
U=ones(1,nzeros);Uc=[ones(1,nzeros),Uc];
Noise = 1/factor*randn(1,maxtime+nzeros);
P=[100 0 0 0;0 100 0 0;0 0 1 0;0 0 0 1]; THETA_hat(:,1)=[-a1 -a2 b0 b1]';beta=[];

alpha = 0.5; gamma = 1.2;
for i=1:maxtime;
   phi=[]; t=i+nzeros; time(t)=i;
   Y(t)=[-A(2) -A(3) B(2) B(3)]*[Y(t-1) Y(t-2) U(t-1) U(t-2)]' + Noise(t-1) + Noise(t-2);
   Ym(t)=[-Am(2) -Am(3) Bm(2) Bm(3)]*[Ym(t-1) Ym(t-2) Uc(t-1) Uc(t-2)]';
```

Algorithm A1 *Cont.*

```
    BETA=(Am(1)+Am(2)+Am(3))/(b0+b1); beta=[beta BETA];

    %RLS implementation
    phi=[Y(t-1) Y(t-2) U(t-1) U(t-2)]'; K=P*phi*1/(lambda+phi'*P*phi); P=P-P*phi*inv(1+phi'*P*phi)*phi'*P/lambda; %RLS-EF
    error(i)=Y(t)-phi'*THETA_hat(:,i); THETA_hat(:,i+1)=THETA_hat(:,i)+K*error(i);
    a1=-THETA_hat(1,i+1);a2=-THETA_hat(2,i+1);b0=THETA_hat(3,i+1);b1=THETA_hat(4,i+1);
    Af(:,i)=[1 a1 a2]'; Bf(:,i)=[b0 b1]';

    % Determine R,S, & T for CONTROLLER
    r1=(b1/b0)+(b1^2-am1*b0*b1+am2*b0^2)*(-b1+a0*b0)/(b0*(b1^2-a1*b0*b1+a2*b0^2));
    s0=b1*(a0*am1-a2-am1*a1+a1^2+am2-a1*a0)/(b1^2-a1*b0*b1+a2*b0^2)+b0*(am1*a2-a1*a2-a0*am2+a0*a2)/(b1^2-a1*b0*b1+a2*b0^2);
    s1=b1*(a1*a2-am1*a2+a0*am2-a0*a2)/(b1^2-a1*b0*b1+a2*b0^2)+b0*(a2*am2-a2^2-a0*am2*a1+a0*a2*am1)/(b1^2-a1*b0*b1+a2*b0^2);
     R=[1 r1];S=[s0 s1];T=BETA*[1 a0];

    Rmatrix=[Rmatrix r1];

     %calculate control signal
    U(t)=[T(1) T(2) -R(2) -S(1) -S(2)]*[Uc(t) Uc(t-1) U(t-1) Y(t) Y(t-1)]';
    U(t)=1.3*[T(1) T(2) -R(2) -S(1) -S(2)]*[Uc(t) Uc(t-1) U(t-1) Y(t) Y(t-1)]';% Arbitrarily increased to duplicate text
end

%store values of control, output, and theta for this estimation method
plotu = [U];
ploty = [Y];
plottheta = [THETA_hat];
%%%%%%%%%%%%%%%%%%%%%%%%%END OF RECURSIVE LEAST SQUARES%%%%%%%%%%%%%%%%%%%%%%%%%
```

Algorithm A2 Autoregressive moving average

```
%%%%%%%%%%%%%%%%%%%%%%%%%AUTOREGRESSIVE MOVING AVERAGE%%%%%%%%%%%%%%%%%%%%%%%%%

B=[0 0.1065 0.0902];A=poly([1.1 0.8]);
H=tf(B,A,0.5);
a1=0;a2=0;b0=0.01;b1=0.2;
Am=poly([0.2+0.2j 0.2−0.2j]);Bm=[0 0.1065 0.0902];
am0=Am(1);am1=Am(2);am2=Am(3);a0=0;

Rmatrix=[];

maxtime=200;

n=4;lambda=1;
nzeros=5;time=zeros(1,nzeros);Y=zeros(1,nzeros);Ym=zeros(1,nzeros);
U=ones(1,nzeros);
THETA_hat = zeros(4,maxtime);
THETA_hat(:,1)=[-a1 -a2 b0 b1]';beta=[];

Noise = 1/factor*randn(1,maxtime+nzeros);
epsilon=[zeros(1,nzeros+maxtime)];
n = 8;
P=10000*eye(n);P(1,1)=1000;P(2,2)=100;P(3,3)=100;P(4,4)=10000;P(5,5)=1000;P(6,6)=100;
theta_hat_els = zeros(n,1);
phi=[];
for i=1:maxtime
    t=i+nzeros; time(t)=i;
    Y(t)=[-A(2) -A(3) B(2) B(3)]*[Y(t-1) Y(t-2) U(t-1) U(t-2)]' + Noise(t-1) + Noise(t-2); %Create truth output
    BETA=(Am(1)+Am(2)+Am(3))/(b0+b1); beta=[beta BETA];
```

Algorithm A2 *Cont.*

```
    phi=[phi; Y(t-1) Y(t-2) U(t-1) U(t-2)];
    Y(t)=[-A(2) -A(3) B(2) B(3)]*[Y(t-1) Y(t-2) U(t-1) U(t-2)]' + Noise(t-1) + Noise(t-2); %Create truth output
    BETA=(Am(1)+Am(2)+Am(3))/(b0+b1); beta=[beta BETA];
    phi=[phi; Y(t-1) Y(t-2) U(t-1) U(t-2)];
    if (i > 3)
        THETA_hat(:,i+1) = inv(phi'*phi)*phi'*Y(1+nzeros:t)';
    else
        THETA_hat(:,i+1) = THETA_hat(:,i);
    end
    a1=-THETA_hat(1,i+1);a2=-THETA_hat(2,i+1);b0=THETA_hat(3,i+1);b1=THETA_hat(4,i+1);% Update A & B coefficients;
    Af(:,i)=[1 a1 a2]'; Bf(:,i)=[b0 b1]'; % Store final A and B for comparison with real A&B to generate epsilon errors
    % Determine R,S, & T for CONTROLLER
    r1=(b1/b0)+(b1^2-am1*b0*b1+am2*b0^2)*(-b1+a0*b0)/(b0*(b1^2-a1*b0*b1+a2*b0^2));
    s0=b1*(a0*am1-a2-am1*a1+a1^2+am2-a1*a0)/(b1^2-a1*b0*b1+a2*b0^2)+b0*(am1*a2-a1*a2-a0*am2+a0*a2)/(b1^2-a1*b0*b1+a2*b0^2);
    s1=b1*(a1*a2-am1*a2+a0*am2-a0*a2)/(b1^2-a1*b0*b1+a2*b0^2)+b0*(a2*am2-a2^2-a0*am2*a1+a0*a2*am1)/(b1^2-a1*b0*b1+a2*b0^2);
    R=[1 r1];S=[s0 s1];T=BETA*[1 a0];

    Rmatrix=[Rmatrix r1];

    %calculate control signal
    U(t)=[T(1) T(2) -R(2) -S(1) -S(2)]*[Uc(t) Uc(t-1) U(t-1) Y(t) Y(t-1)]';
    U(t)=1.3*[T(1) T(2) -R(2) -S(1) -S(2)]*[Uc(t) Uc(t-1) U(t-1) Y(t) Y(t-1)]';% Arbitrarily increased to duplicate text
end

plotu = [plotu; U];
ploty = [ploty; Y];
plottheta = [plottheta; THETA_hat];
%%%%%%%%%%%%%%%%%%%%END OF AUTOREGRESSIVE MOVING AVERAGE%%%%%%%%%%%%%%%%%%%%
```

Algorithm A3 Extended Least Squares

```
%%%%%%%%%%%%%%%%%%%%EXTENDED LEAST SQUARES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B=[0 0.1065 0.0902];A=poly([1.1 0.8]);
H=tf(B,A,0.5);
a1=0;a2=0;b0=0.01;b1=0.2;
Am=poly([0.2+0.2j 0.2−0.2j]);Bm=[0 0.1065 0.0902];
am0=Am(1);am1=Am(2);am2=Am(3);a0=0;

Rmatrix=[];

maxtime=200;

n=4;lambda=1;
nzeros=5;time=zeros(1,nzeros);Y=zeros(1,nzeros);Ym=zeros(1,nzeros);
U=ones(1,nzeros);
THETA_hat(:,1)=[-a1 -a2 b0 b1]';beta=[];% Initialize P(to), THETA_hat(to) & Beta

Noise = 1/factor*randn(1,maxtime+nzeros);
epsilon=[zeros(1,nzeros+maxtime)];
n = 8;
P=10000*eye(n);P(1,1)=1000;P(2,2)=100;P(3,3)=100;P(4,4)=10000;P(5,5)=1000;P(6,6)=100;
theta_hat_els = zeros(n,1);
for i=1:maxtime;
    phi=[]; t=i+nzeros; time(t)=i;
    Y(t)=[-A(2) -A(3) B(2) B(3)]*[Y(t-1) Y(t-2) U(t-1) U(t-2)]' + Noise(t-1) + Noise(t-2); %Create truth output
    Ym(t)=[-Am(2) -Am(3) Bm(2) Bm(3)]*[Ym(t-1) Ym(t-2) Uc(t-1) Uc(t-2)]';
    BETA=(Am(1)+Am(2)+Am(3))/(b0+b1); beta=[beta BETA];
```

Algorithm A3 *Cont.*

```
    k=i+nzeros;
    phi=[Y(t-1) Y(t-2) U(t-1) U(t-2) epsilon(t) epsilon(t-1) epsilon(t-2) epsilon(k-3)]';
    K=P*phi*1/(1+phi'*P*phi);
     P=P-P*phi*pinv(1+phi'*P*phi)*phi'*P;
    error(i)=Y(k)-phi'*theta_hat_els(:,i);
    theta_hat_els(:,i+1)=theta_hat_els(:,i)+K*error(i);
    epsilon(k)=Y(k)-phi'*theta_hat_els(:,i+1); %Form Posterior Residual

    THETA_hat(:,i+1) = theta_hat_els(1:4,i+1);
    a1=-THETA_hat(1,i+1);a2=-THETA_hat(2,i+1);b0=THETA_hat(3,i+1);b1=THETA_hat(4,i+1);% Update A & B coefficients;
    Af(:,i)=[1 a1 a2]'; Bf(:,i)=[b0 b1]'; % Store final A and B for comparison with real A&B to generate epsilon errors

    r1=(b1/b0)+(b1^2-am1*b0*b1+am2*b0^2)*(-b1+a0*b0)/(b0*(b1^2-a1*b0*b1+a2*b0^2));
    s0=b1*(a0*am1-a2-am1*a1+a1^2+am2-a1*a0)/(b1^2-a1*b0*b1+a2*b0^2)+b0*(am1*a2-a1*a2-a0*am2+a0*a2)/(b1^2-a1*b0*b1+a2*b0^2);
    s1=b1*(a1*a2-am1*a2+a0*am2-a0*a2)/(b1^2-a1*b0*b1+a2*b0^2)+b0*(a2*am2-a2^2-a0*am2*a1+a0*a2*am1)/(b1^2-a1*b0*b1+a2*b0^2);
    R=[1 r1];S=[s0 s1];T=BETA*[1 a0];

    Rmatrix=[Rmatrix r1];

    %calculate control signal
    U(t)=[T(1) T(2) -R(2) -S(1) -S(2)]*[Uc(t) Uc(t-1) U(t-1) Y(t) Y(t-1)]';
    U(t)=1.3*[T(1) T(2) -R(2) -S(1) -S(2)]*[Uc(t) Uc(t-1) U(t-1) Y(t) Y(t-1)]';% Arbitrarily increased to duplicate text
end

plotu = [plotu; U];
ploty = [ploty; Y];
plottheta = [plottheta; THETA_hat];
%%%%%%%%%%%%%%%%%%%%%%%%%%%END OF EXTENDED LEAST SQUARES%%%%%%%%%%%%%%%%%%%%%%
```

Algorithm A4 Deterministic Artificial Intelligence

```
%%%%%%%%%%%%%%%%%%%%%%DETERMINISTIC
AI%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B=[0 0.1065 0.0902];A=poly([1.1 0.8]);
H=tf(B,A,0.5); %Convert Plant [num] and [den] to discrete transfer function
Rmatrix=[];

%Create command signal, Uc based on Example 3.5 plots...square wave with 50 sec period

n=4;lambda=1; % number of parameters to estimate and Exponential Forgetting Factor
nzeros=5;time=zeros(1,nzeros);Y=zeros(1,nzeros);Ym=zeros(1,nzeros);%Initialize ouput vectors
U=ones(1,nzeros);

Noise = 1/25*randn(1,maxtime+nzeros);
epsilon=[zeros(1,nzeros+maxtime)];
n = 4;

phi_awr = [];
ustar = [];
hatvec = [];

t=[0:200];

hvy_m = [zeros(1,nzeros) traj_Uc];

eb = Y(1) - hvy_m(1);
```

Algorithm A4 *Cont.*

```
err = 0;

kp = 2.0;
kd = 6.0;
phid = [];
ustar = [];
hatvec = zeros(4,1);
for i=1:maxtime+1; %Loop through the output data Y(t)
    t=i+nzeros; time(t)=i;
    de = err-eb;
    u = kp*err + kd*de;
    U(t-1) = u;
    Y(t)=[-A(2) -A(3) B(2) B(3)]*[Y(t-1) Y(t-2) U(t-1) U(t-2)]' + Noise(t-1) + Noise(t-2);
    phid = [phid; Y(t) -Y(t-1) Y(t-2) -U(t-2)];
    ustar = [ustar; u];
    newest = phid\ustar;
    hatvec(:,i) = newest;
    eb = err;
    disp(t);
    err = hvy_m(t)-Y(t);
end

THETA_hat = [hatvec(2,:)./hatvec(1,:); hatvec(3,:)./hatvec(1,:); ones(1,201)./hatvec(1,:); hatvec(4,:)./hatvec(1,:)];
plotu = [plotu; U];
ploty = [ploty; Y(1:205)];
plottheta = [plottheta; THETA_hat];

%%%%%%%%%%%%%%%%%%%%%%%%%%END OF DETERMINISTIC AI%%%%%%%%%%%%%%%%%%%%%%%%%%
```

References

1. Liu, Z.; Zhuang, X.; Wang, S. Speed Control of a DC Motor using BP Neural Networks. In Proceedings of the 2003 IEEE Conference on Control Applications, Istanbul, Turkey, 25–25 June 2003; pp. 832–835.
2. Mishra, M. Speed Control of DC Motor Using Novel Neural Network Configuration. Bachelor's Thesis, National Institute of Technology, Rourkela, India, 2009.
3. Hernández-Alvarado, R.; García-Valdovinos, L.G.; Salgado-Jiménez, T.; Gómez-Espinosa, A.; Fonseca-Navarro, F. Neural Network-Based Self-Tuning PID Control for Underwater Vehicles. *Sensors* **2016**, *16*, 1429. [CrossRef] [PubMed]
4. Rashwan, A. An Indirect Self-Tuning Speed Controller Design for DC Motor Using A RLS Principle. In Proceedings of the 21st International Middle East Power Systems Conference (MEPCON), Cairo, Egypt, 17–19 December 2019; pp. 633–638.
5. Sands, T. Development of deterministic artificial intelligence for unmanned underwater vehicles (UUV). *J. Mar. Sci. Eng.* **2020**, *8*, 578. [CrossRef]
6. Sands, T. Control of DC Motors to Guide Unmanned Underwater Vehicles. *Appl. Sci.* **2021**, *11*, 2144. [CrossRef]
7. Keller, J. Navy Eyes Unmanned Underwater Vehicle (UUV) Weapons Payloads to Stop or Disable 160-Foot Ships at Sea. 24 May 2018. Available online: https://www.militaryaerospace.com/unmanned/article/16726886/navy-eyes-unmanned-underwater-vehicle-uuv-weapons-payloads-to-stop-or-disable-160foot-ships-at-sea (accessed on 16 April 2021).
8. Rees, C. Maxon Launches High Torque DC Brushless Motors. 5 May 2015. Available online: https://www.unmannedsystemstechnology.com/2015/05/maxon-launches-high-torque-dc-brushless-motors/ (accessed on 16 April 2021).
9. Underwater Thruster Propeller Motor for ROV AUV. Available online: https://www.alibaba.com/product-detail/underwater-thruster-propeller-motor-for-ROV_62275939884.html (accessed on 16 April 2021).
10. Gauss, C. Theoria Motus Corporum Coelestum Werke, 1809. In *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Section*; Davis, C.H., Translator; Dover: New York, NY, USA, 1963. Available online: https://doi.org/10.5962/bhl.title.19023 (accessed on 27 May 2021).
11. Gauss, C. Theoria Combinationis Erroribus. 1821, 1823a, 1826. In *Theory of the Combination of Observations Least Subject to Errors*; Stewart, G.W., Translator; SIAM: Philadelphia, PA, USA, 1995.
12. Gauss, C. *Minimis Obnoxiae, Parts 1, 2, and Supplement Werke 4, l-108, ~1803–1809 Disquisitiones de elementis ellipticis Pallidis Werke 6, l–24*; Reprinted by Werke: Leipzig–Berlin, 1863–1933; Cambridge University Press: Cambridge, UK, 2013. Available online: https://doi.org/10.1017/CBO9781139058247 (accessed on 27 May 2021).

13. Hamel, J. Heinrich Christian Schumacher-mediator between Denmark and Germany Center of Scientific Communication in Astronomy. In *Around Caspar Wessel and the Geometric Representation of Complex Numbers: Proceedings of the Wessel Symposium at the Royal Danish Academy of Sciences and Letters, Copenhagen, 11–15 August 1998*; Invited Papers; Videnskabernes Selskab: Copenhagen, Denmark, 2001; pp. 99–120. Available online: http://gymarkiv.sdu.dk/MFM/kdvs/mfm%2040-49/mfm-46-2.pdf (accessed on 29 April 2021).
14. Astrophysical Institute. *Astronomische Nachrichten*; Archived from the original on 28 June 2012; Astrophysical Institute: Potsdam, Germany, 2012. Available online: https://onlinelibrary.wiley.com/loi/15213994/year/1823 (accessed on 29 April 2021).
15. Kalbfleisch, J. *A Source Book of Mathematics*; McGraw-Hill Book Company: New York, NY, USA, 1929; pp. 576–579.
16. Legendre, A. *Notes on Nouvelles Methodes pour la Determination des Orbites des Cometes*; Second supplement to the third edition of Legendre (1805); Stigler; Ruger Legendre, H.A.; Walker, H.M., Translators; Courcier: Paris, France; McGraw-Hill: New York, NY, USA, 1920. Available online: https://catalogue.nla.gov.au/Record/866184 (accessed on 27 May 2021).
17. Åström, K.; Wittenmark, B. On the Control of Constant but Unknown Systems. In Proceedings of the 5th IFAC World Congress, Paris, France, 12–17 June 1972.
18. Åström, K.; Wittenmark, B. On self-tuning regulators. *Automatica* **1973**, *9*, 185–199. [CrossRef]
19. Åström, K.; Wittenmark, B. *Adaptive Control*; Addison-Wesley: Boston, FL, USA, 1995; pp. 43, 48, 63–65, 331.
20. Sands, T. Space System Identification Algorithms. *J. Space Explor.* **2018**, *6*, 138.
21. Sands, T. Nonlinear-Adaptive Mathematical System Identification. *Computation* **2017**, *5*, 47. [CrossRef]
22. Slotine, J.; Weiping, L.W. *Applied Nonlinear Control*; Prentice-Hall: London, UK, 1991; pp. 331, 365.
23. Fossen, T. Comments on Hamiltonian adaptive control of spacecraft by J.J.E. Slotine and M.D. Di Benedetto. *IEEE Trans. Autom. Control* **1993**, *38*, 671–672. [CrossRef]
24. Sands, T.; Kim, J.; Agrawal, B. Improved Hamiltonian Adaptive Control of spacecraft. In Proceedings of the 2009 IEEE Aerospace Conference, Big Sky, MT, USA, 7–14 March 2009; pp. 1–10. [CrossRef]
25. Shorey, T.; Tijdeman, R. Exponential Diophantine Equations. In *Cambridge Tracts in Mathematics*; Cambridge University Press: Cambridge, UK, 1986.
26. Bézout, É. *Theorie Generale Des Equations Algebriques (1779)*; Kessinger Publishing: Whitefish, MT, USA, 2010.
27. Bhau, D. Brief Notes on the Age and Authenticity of the Works of Aryabhata, Varahamihira, Brahmagupta, Bhattotpala, and Bhaskaracharya. *J. R. Asiat. Soc. G. B. Irel.* **1865**, *1*, 392–418.
28. Schoukens, M.; Noël, J.P. Three Benchmarks Addressing Open Challenges in Nonlinear System Identification. In Proceedings of the 20th World Congress of the International Federation of Automatic Control, Toulouse, France, 9–14 July 2017; pp. 448–453.
29. Sands, T.; Kim, J.; Agrawal, B. Spacecraft fine tracking pointing using adaptive control. In Proceedings of the 58th International Astronautical Congress, Hyderabad, India, 24–28 September 2007; International Astronautical Federation: Paris, France, 2007.
30. Sands, T.; Lorenz, R. Physics-Based Automated Control of Spacecraft. In Proceedings of the AIAA Space Conference & Exposition, Pasadena, CA, USA, 14–17 September 2009.
31. Sands, T.; Kim, J.J.; Agrawal, B.N. Spacecraft Adaptive Control Evaluation. In Proceedings of the Infotech@ Aerospace, Garden Grove, CA, USA, 19–21 June 2012; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2012; pp. 2012–2476.
32. Smeresky, B.; Rizzo, A.; Sands, T. Optimal Learning and Self-Awareness versus PDI. *Algorithms* **2020**, *13*, 23. [CrossRef]
33. Guida, D.; Nilvetti, F.; Pappalardo, C.M. Parameter Identification of a Two Degrees of Freedom Mechanical System. *Int. J. Mech.* **2009**, *3*, 23–30.
34. Guida, D.; Pappalardo, C.M. Sommerfeld and Mass Parameter Identification of Lubricated Journal Bearing. *WSEAS Trans. Appl. Theor. Mech.* **2009**, *4*, 205–214.
35. Heidlauf, P.; Cooper, M. Nonlinear Lyapunov Control Improved by an Extended Least Squares Adaptive Feed Forward Controller and Enhanced Luenberger Observer. In Proceedings of the International Conference and Exhibition on Mechanical & Aerospace Engineering, Las Vegas, NV, USA, 2–4 October 2017.
36. Baker, K.; Cooper, M.; Heidlauf, P.; Sands, T. Autonomous Trajectory Generation for Deterministic Artificial Intelligence. *Electr. Electron. Eng.* **2018**, *8*, 59–68. [CrossRef]
37. Sands, T. Comparison and Interpretation Methods for Predictive Control of Mechanics. *Algorithms* **2019**, *12*, 232. [CrossRef]
38. Marusak, P.M. Numerically Efficient Fuzzy MPC Algorithm with Advanced Generation of Prediction—Application to a Chemical Reactor. *Algorithms* **2020**, *13*, 143. [CrossRef]
39. Marusak, P.M. Advanced Construction of the Dynamic Matrix in Numerically Efficient Fuzzy MPC Algorithms. *Algorithms* **2021**, *14*, 25. [CrossRef]
40. Nebeluk, R.; Ławryńczuk, M. Tuning of Multivariable Model Predictive Control for Industrial Tasks. *Algorithms* **2021**, *14*, 10. [CrossRef]
41. Pappalardo, C.; Guida, D. On the dynamics and control of underactuated nonholonomic mechanical systems and applications to mobile robots. *Arch. Appl. Mech.* **2019**, *89*, 669. [CrossRef]
42. Pappalardo, C.M.; Guida, D. System Identification Algorithm for Computing the Modal Parameters of Linear Mechanical Systems. *Machines* **2018**, *6*, 12. [CrossRef]
43. Bruzzone, L.; Fanghella, P.; Baggetta, M. Experimental Assessment of Fractional-Order PDD1/2 Control of a Brushless DC Motor with Inertial Load. *Actuators* **2020**, *9*, 13. [CrossRef]