Article

Autonomous Drone Electronics Amplified With Pontryagin–Based Optimization

Jiahao Xu and Timothy Sands *

Posted Date: 3 April 2023

doi: 10.20944/preprints202304.0010.v1

Keywords: Systems Engineering; Path Planning; deterministic artificial intelligence; System Identification; DC-motor; Least Square; Autonomous Trajectory Generation

Article

Autonomous Drone Electronics Amplified with Pontryagin-Based Optimization

Jiahao Xu [1] and Timothy Sands [2,*]

[1] Sibley School of Mechanical and Aerospace Engineering, Cornell University, USA; jx98@cornell.edu
[2] Department of Mechanical Engineering (SCPD), Stanford University, Stanford 94305, USA
* Correspondence: dr.timsands@alumni.stanford.edu

Abstract: In the era of electrification and artificial intelligence, direct current motors are widely utilized with numerous innovative adaptive and learning methods. Traditional methods utilize model-based algebraic techniques with system identification, such as recursive least squares, extended least squares, and autoregressive moving averages. The new method known as deterministic artificial intelligence asserts physical-based process dynamics to achieve target trajectory tracking. There are two common autonomous trajectory generation algorithms: sinusoidal function and Pontryagin's based generation algorithm. This thesis aims to simulate model-following and deterministic artificial intelligence methods using sinusoidal and Pontryagin's methods and compare their performance difference when following challenging step function slew maneuver.

Keywords: systems engineering; path planning; deterministic artificial intelligence; system identification; DC-motor; least square; autonomous trajectory generation

1. Introduction

1.1. Motivation

As electrification, control theory, and advanced manufacturing technology continue to evolve. The National Aeronautics and Space Administration (NASA) identified unmanned aerial vehicle (UAV) technology as a critical area for future development, as illustrated in Figure 1. Adaptive and learning systems have significantly broadened the operational designed domain (ODD) of UAVs, enabling them to function with minimal human intervention and rapidly self-tune during unpredictable weather, extreme working conditions, or even partially damaged conditions.

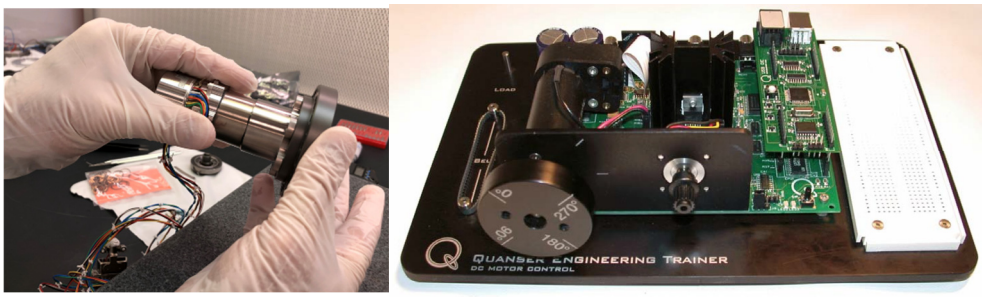
(**a**) (**b**)

Figure 1. Drones in flight in downtown Reno, Nevada, during shakedown tests for NASA's Unmanned Aircraft Systems Traffic Management project [1]. (**b**) A drone flies above Lake Street in downtown Reno, Nevada, for UTM's street-level testing during the so-called technical capability level event #4, on May 20, 2019 [2]. Image credits: NASA/Ames Research Center/Dominic Hart used in compliance with image use policy [3].

Direct current (DC) motors, illustrated in Figure 2, are a type of rotary electrical motors that convert direct current electrical energy into mechanical energy. DC motors are the initial type of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

motors that gained widespread usage due to their ability to be powered by existing direct-current lighting power distribution systems. As a result, they are widely used as the driving mechanism for UAV propellers.



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

Figure 2. (**a**) DC motor prior to integration [4] image credit Motiv Space Systems, Inc. Images used in compliance with image use policy [3]. (**b**) Quanser DC motor control [5,6].

Control of DC motors is a classic topic with recent novel methods such as deterministic artificial intelligence (DAI) presented here, neural networks [7–12], and reinforcement learning [13] and other instantiations of neural network-based, stochastic artificial intelligence [14–16].

Khomenko et. al, [9] sought optimal control of motor positioning using a combination of artificial neural network and state space method with variable gain yielding a transient process close to optimal without overshoot. The approach proposed in this manuscript has similar efficacies. gesh, et. al, [10] investigated dual implementation: a neural estimator used to estimate the motor speed and a neural controller used to generate a control signal for a converter, illustrating effectiveness and advantages in comparison with conventional control schemes. Naung, et. al, [11] illustrated improved speed and torque dynamic responses of DC motor by using neural network parameter tuner with a classical proportional, integral controller. Yang et. al, [12] proposed neural networks with output feedback to deal with DC motor measurement noise and unknown dynamics including friction, parametric uncertainties, external disturbances and unmodeled dynamics. Ćirić et. al, [13] applied an adequate classifier based on a deep neural network achieving of about 87% using only 668 data samples. Nizami et, al. [14] proposed a single functional layer Legendre neural network integrated adaptive backstepping control technique and favorably compared the performance to the response obtained from proportional-integral-derivative controller. Lei, et. al, [15] proposed self-tuning and approximation via RBF neural networks, validating control system accuracy owing to its robustness and adaptability. Seeking to avoid corrective maintenance to reduce costs, Scalabrini, et. al, [16] used an artificial neural network to prediction of motor failure time.

Just this year, Zhang, et. al, [17] appended a virtual DC motor control to the converter to effectively suppress the fluctuation of the DC bus reducing the rate to 9%, and the voltage recovery time is only 0.18 seconds. Tufenkci, et. al, [18] proposed reinforcement learning of PI control dynamics for optimal speed control of DC motors by using Twin Delay Deep Deterministic Policy Gradient Algorithm. Munagala, et. al, [19] proposed controlling DC motor speed using a technique for identifying the system dynamics for neural network–based fractional order proportional integral derivative controller. Highlighting computer numerical control machine DC motors are still affected by transmission torque ripple, which mostly depends on the speed and the transient line current at the transmission interval, Prakash, et. al, [20] utilized a combination of so-called golden eagle optimization and radial basis function neural network to reduce ripple to 1.26%. Ghany, et. al, [21] illustrated an efficient interval type-2 fuzzy-based, single neuron proportional–integral–derivative controller can improve dynamic motor response accommodating system uncertainty. Baidya, et. al, [22] utilized a sensor feedback-supported controller based on a novel dandelion optimization PID controller could produce integral square errors, integral absolute errors, integral errors, integral time square errors, and integral time absolute errors on the order of 10^{-2}. Sorfina, et. al, [23] illustrated the impacts of deleterious performances of DC motors resulted in 14% failure rate when utilized for

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

photovoltaic cleaning system automation. Mohanraj, et. al, [24] illustrated many advantages including independent variable speed and variable torque operation along with regeneration capability when using DC motor-based electric drive systems by including road friction, aerodynamic forces and transmission, are considered for calculating the motor shaft torque using model predictive control and proportional integral control, while only being able to assert the scheme makes significant energy capture possible. Thus continued, improvements that may be validated are continually developing. Yang, et. al, [26] proposed efficiency as much as 15–20% higher is possible with sliding-mode based PID control, while merely asserting effectiveness. 27. Tripathi, et. al, [27] proposed a quite novel fractional order adaptive Kalman filter for sensor–less motor speed control, but merely achieved robustness and accuracy improvements in state estimation in comparison with extended Kalman filter instantiations. Saini, et. al, [28] claim better performance using an enhanced hybrid stochastic fractal search controller. Rahman, et. al, [29] also tried a fractional transformation but implemented such on an H-infinity controller. The results of that 2023 study are displayed in Figure 3 and should be considered benchmark for comparison of the results presented in this manuscript (noting the undershoot and overshoot characteristics).

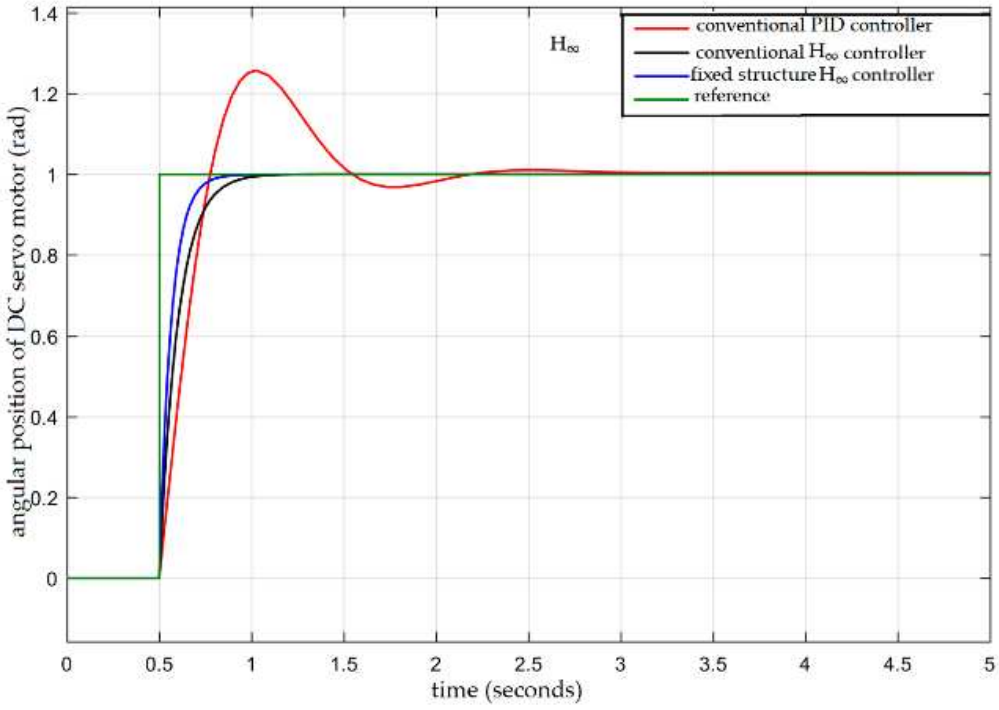


Figure 3. Benchmark 2023 study by Rahman, et. al, [29] using a fractional transformation implemented on an H-infinity controller.

Highlighting the vulnerability of conventional direct torque control to ripple, Kumar, et. al, [30] sought minimization and speed regulation, where their novel instantiation of space vector pulse width modulation reduced the toque ripple by 63.1% when compared to conventional PI and 58.5% when compared to a second benchmark, a PSO-PI controller.

Mérida-Calvo, et. al, [31] also focused on commanded trajectory tracking seeking to address both time-delays (tracking lag) and overshoots. They assert PID controllers do not yield a good tracking performance owing to friction nonlinearity and proposed adding a prefilter combined with a Smith predictor, an anti-windup scheme and a Coulomb friction compensator. The performance achieved is presented in Figure 4.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

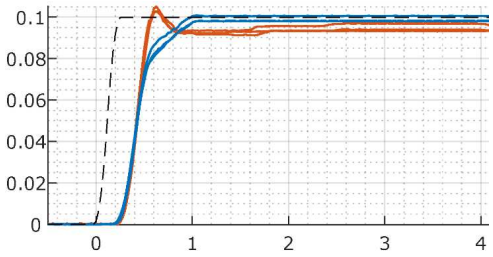


Figure 4. Benchmark 2023 study by Mérida-Calvo, et. al, [31] seeking to address both time-delays (tracking lag) and overshoots by focusing on commanded trajectory tracking.

Vered, et. al, [32] used digital twins to remotely update feedback controllers and the results produced very accurate trajectory tracking of square wave commands (as displayed in Figure 5, while the overshoots and settling (also called "ringing") remain a residual concern. The results just achieved by Vered in 2023 are an available high-performance benchmark for comparing the results presented in this manuscript, where such tight target-tracking is sought without the "ringing".

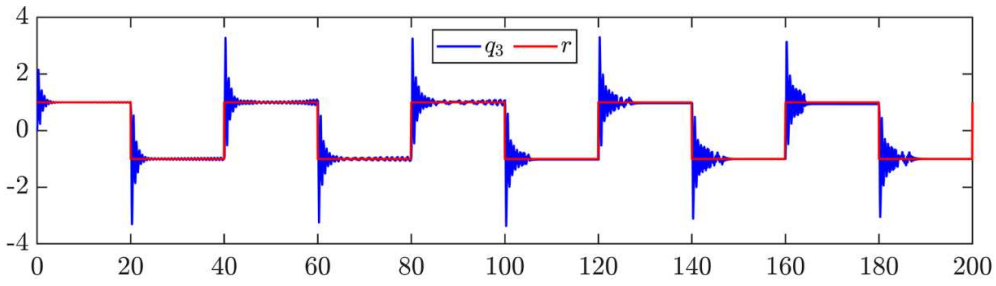


Figure 5. Modestly accurate square wave trajectory tracking benchmark by Vered, et. al, [32] using digital twins to remotely update feedback controllers. Note overshoots and settling remain.

Improved performance was achieved by Gurumoorthy, et. al, [33], and the results modestly mimic the results presented in this manuscript, albeit the delirious efficacies of noise remain prevalent.

Perhaps the proposals closest to those presented in this manuscript come from Stanford's Moehle and Boyd [36] whose methods most closely mimic the proposed work, following the Cornell University lineage of research, where comparisons are made of sinusoidal versus optimal instantiations akin the deterministic artificial intelligence methods to be proposed in this manuscript. Figures 6 and 8 in [36] illustrate the difficulty overcoming back electromagnetic force while attempting to follow a square wave. Very similar results were presented by Stanford's Niemeyer et. al, [37,38] and later by colleague Diolaiti [39], illustrating the temporal longevity of the challenge identifying cancellation-replacement approaches experience performance limits due to sensor quantization, discretization, and amplifier bandwidths, where virtual stiffness was introduced to address these limitations.

On the other hand, deterministic artificial intelligence proposed in this manuscript utilize self-awareness statements that are deterministic in nature to establish governing differential equations based on either the underlying physical properties of the problem or system identification methods [40–50]. This characteristic distinguishes it from the commonly used stochastic approaches in artificial intelligence, as it primarily employs first principles whenever feasible. Utilizing deterministic artificial intelligence principles instead of commonly used methods has the advantage of a simpler approach once the re-parameterization is derived. Despite being deterministic, its optimality in terms of self-awareness and learning can be readily understood by researchers who possess basic knowledge of linear regression.

Bernat, et. al, [40] proposed model-reference adaptive control, while Gowri, et. al, [41] proposed direct induction motor torque control using discontinuous pulse width modulation algorithm

seeking to reduce current ripples. Rathaiah, et. al, [42] optimized the adaptive control method, while Haghi, et. al, [43] made similar proposals using extremum seeking methods. These approaches were paralleled with a predominant lineage of physics-based work by the research group of Lorenz at the University of Wisconsin [44], and the efforts continue [45–49]. Zhang, et. al, [45] used an analogous three-phase motor illustrating efficacies applied to direct torque control of five phase motors. Apoorva, et. al, [46] extended Lorenz's physics-based techniques to variable flux motors. Flieh, et. al, [47] illustrated loss minimization for various servo motors, in permanent magnet motor systems [48], and self-sensing via flux injection [49] with parallel application by Vidlak, et. al, illustrating ripple control [50]. The deterministic artificial intelligence proposed here adopts the physics-based feedforward approaches of Lorenz's group, and that adoption necessitates prescribed trajectories.

When applying deterministic artificial intelligence, autonomous trajectory generation techniques are needed to approximate non-differentiable transient changes using a smooth curve. Two of the most used techniques for this purpose are the sinusoidal and Pontryagin's methods. The latter involves utilizing boundary conditions to compute optimal trajectories based on control cost and will be compared to the sinusoidal method, which is a simpler technique for trajectory generation.

This manuscript expands upon the analysis of deterministic artificial intelligence as presented in the following literature review, with the main purpose of advocating for the commercial use of deterministic artificial intelligence in unmanned vehicles. The primary content of this manuscript consists of a detailed comparison of both discrete deterministic artificial intelligence and a selected state-of-the-art benchmark approach, with a specific focus on their trajectory-tracking capabilities when combined with different trajectory generation methods. The comparison aims to provide a comprehensive evaluation of the strengths and weaknesses of each approach in the context of trajectory tracking, thereby aiding in a more informed decision-making process when selecting a suitable methodology for unmanned aerial vehicle applications.

1.2. State of the art benchmarks

The following list highlights the current state of the art developing deterministic artificial intelligence:

1. In 2021, reference [51] illustrated reveals that deterministic artificial intelligence yields 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).
2. That same year, seeking to duplicate the results, Shah [52] discerned 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 was declared surprising and not in accordance with the recently published literature, and the explanation of the difference was theorized to be efficacy with discretized implementations.
3. The following year, in 2022 Koo, et. al, [53] in response to Shah's recommended future research investigated the impacts of discretization (timestep) and numerical propagation on the deterministic artificial intelligence approach.
4. In 2023, Menezes, et. al, [54] investigated the residual feature of discretization methods to complement the work of Koo, where the first order hold method of discretization with a surprisingly large sample time of seven-tenths of second yields greater than sixty percent improvement over the results presented in the prequel literature. Seemingly deterministic artificial intelligence might be less susceptible to larger step sizes when using first order hold discretization.
5. Subsequently in 2023, Wang, et, al. [55] investigated necessary step sizes for discrete applications to approach the performance of the continuous application of deterministic artificial intelligence. The revealed error means were roughly approximate when the step size was reduced to 0.2 seconds.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

1.3. Novelties presented

The following list presents the novelties presented in this manuscript towards the development of deterministic artificial intelligence.

1. Five of the most used DC motor model–following methods are presented.
2. Autonomous trajectory generation methods using Pontryagin's method will be proposed.
3. Comparisons are offered of deterministic artificial intelligence and the state-of-the-art model following method(s) using different trajectory generation methods and the superiority of discrete deterministic artificial intelligence augmented with Pontryagin's method is revealed.
4. Comparison is offered for continuous deterministic artificial intelligence with different trajectory generation methods and a possible breaking point in engineering applications (smallest error but big input value, may exceed the max input for the DC motor) is revealed.

2. Materials and Methods

DC motors are well-studied by many methods, providing good comparative benchmarks. Section 2.1 elaborates on motor modeling. The canonical motor model from [56] with bump-test current regulation is integrated for voltage display, where the bump-test reasonably resembles each discontinuous jump of the square wave. The canonical model is available for purchase by readers seeking to repeat the work in this manuscript, and its designation is Quanser USB QICii, and the canonical motor is equation 2.6 in the laboratory workbook DC Motor Control Trainer (DCMCT), where section 2.6.1.1 [56]. Module Description of the workbook elaborates the modeling methods for rate output, The workbook's author is the same author of [57] where the same motor model is assumed, which provides the benchmark methods in this manuscript, e.g., provides the starting point. This is important to allow researchers to duplicate these results.

2.1. Motor modeling

In order to accurately describe the transfer function of a DC motor, it is necessary to use a continuous-time process and a normalized model to determine the voltage for the equivalent circuit model. This voltage is dependent on the current change at the operating point, which is in turn determined by the state of charge. By using Equation (1), the transfer function can be expressed in a clear and concise manner.

$$G(s) = \frac{B(s)}{A(s)} = \frac{1}{s(s+1)} \tag{1}$$

The discrete-time signal's frequency domain depiction is shown in Equation (2). To obtain this, the signal was discretized using the zero-pole matching method with an initial continuous-time process of 0.35 seconds, implemented through the MATLAB® function c2d. The control signal and output in discrete time are denoted by U(z) and Y(z) respectively, using the z-transform. Equation (3) presents the resulting differential equation, along with the relevant variables and nomenclature defined in Table 1.

$$G(z) = \frac{Y(z)}{U(z)} = \frac{0.0517z + 0.0517}{z^2 - 1.7047z + 0.7047} \tag{2}$$

$$0.0517u(t) + 0.0517u(t-1) = y(t+1) - 1.7047y(t) + 0.7047y(t-1) \tag{3}$$

Table 1. Table of proximal variables and nomenclature [1].

Variable/acronym	Definition	Variable/acronym	Definition
G	Transfer function	s	Differential variable
Y	Output	z	Difference variable
U	Input	t	Discrete time variable

[1] Such tables are offered throughout the manuscript to aid readability.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

To achieve stability, the model-following method can be applied to relocate the unstable pole present in the transfer function of the process to stable positions. However, this technique differs from deterministic artificial intelligence modeling which does not emphasize pole relocation. Rather, it employs a proportional plus derivative (PD) feedback adaptation of unknown parameters to achieve autonomous tracking of the desired trajectory.

2.2. Model-Following Self Tuner

A control system was established in this study, which employed a dynamic feedforward for the input u_c and dynamic feedback for the output y. These were then combined to generate the process input u, as shown in Figure 3.3 in [57]. The model-following topology adopted in this study can be compared to the deterministic artificial intelligence topology depicted in Figure 2 in [57]. The system's response can be described using Equations (4) and (5), where U(z) and Y(z) correspond to the z-transform of the control signal and control output, respectively.

$$Y(z) = \frac{B}{A}\left(\frac{T}{R}U(z) - \frac{S}{R}Y(z)\right) \tag{4}$$

$$G(z) = \frac{Y(z)}{U(z)} = \frac{BT}{AR + BS} \tag{5}$$

To obtain the desired transfer function of the system, cancellations are built into Equation (6), resulting in the numerator and denominator being factorized into several parts. The canceled zeros are represented by B^+, while the uncancelled zeros are represented by B^-. B'_m represents a scalar multiple of the system, and A_0 represents pole-zero cancellations. The factorization of the numerator and denominator into these parts enables us to obtain the desired transfer function of the system.

$$\frac{Y(z)}{U(z)} = \frac{B^+B^-A_0B'_m}{A_0A_mB^+} = \frac{BT}{AR + BS} = \frac{B_m}{A_m} \tag{6}$$

In order to ensure that the causality conditions are met, the polynomials R, S, and T are constrained to first order, and B^+ is set to 1 since there will be no process zero cancellations. The coefficients of the polynomials R', S, and T can be determined by manipulating Equation (6), while Equations (7)-(9) from [57] provide a way to express these coefficients in terms of the desired process parameters and their estimated values.

$$r_1 = \frac{b_1}{b_0} + \frac{\left(b_1^2 - a_{m_1}b_0b_1 + a_{m_2}b_0^2\right)(-b_1 + a_0b_0)}{b_0(b_1^2 - a_1b_0b_1 + a_2b_0^2)} \tag{7}$$

$$s_0 = \frac{b_1\left(a_0a_{m_1} - a_2 - a_{m_1}a_1 + a_1^2 + a_{m_2} - a_1a_0\right)}{b_1^2 - a_1b_0b_1 + a_2b_0^2} + \frac{b_0\left(a_{m_1}a_2 - a_1a_2 - a_0a_{m_2} + a_0a_2\right)}{b_1^2 - a_1b_0b_1 + a_2b_0^2} \tag{8}$$

$$s_1 = \frac{b_1\left(a_1a_2 - a_{m_1}a_2 + a_0a_{m_2} - a_0a_2\right)}{b_1^2 - a_1b_0b_1 + a_2b_0^2} + \frac{b_0\left(a_2a_{m_2} - a_2^2 - a_0a_{m_2}a_1 + a_0a_2a_{m_1}\right)}{b_1^2 - a_1b_0b_1 + a_2b_0^2} \tag{9}$$

In order to combine feedforward and feedback, equations (10) and (11) are used to express the control signal input to the process. Reference [57] emphasizes the importance of the parameter β in driving the system to unity gain, which is a crucial step in achieving zero asymptotic tracking error.

$$RU(z) = TU_c(z) + SY(z) \tag{10}$$

$$u(t) = \beta U_c(t) + \beta a_0 U_c(t-1) + s_0Y(t) + s_1Y(t-1) - r_1U(t-1) \tag{11}$$

In this article, different parameter estimation methods will be presented and compared to estimate the unknown parameters. The methods that will be discussed are recursive least square (RLS), recursive least square with exponential forgetting (RLSwEF), auto regressive moving average (ARMA), extended least square (ELS), and extended least square with posterior residuals (ELSwPR).

Recursive least square is a widely used method for self-tuning model following, which recursively estimates the unknown parameters of a system by minimizing the squared error between

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

predicted and actual output. Recursive least square with exponential forgetting is an extension that employs a forgetting factor to give more weight to recent data and prevent the algorithm from being overly influenced by past data. Auto regressive moving average is a method that models the process output as a combination of autoregressive and moving average components. Extended least square is another self-tuning method that extends the least square algorithm to estimate the parameters of a system using both input and output data. Extended least square with posterior residuals is a further extension that uses the residuals of the posterior mean estimate as a measure of the quality of the estimate and adaptively adjusts the forgetting factor to achieve better tracking performance. In this article, we present and compare these different parameter estimation methods.

2.3. Deterministic artificial intelligence

In order to establish the concept of self-awareness in the context of deterministic artificial intelligence, it is necessary to rearrange Equation (3) to isolate $u(t)$ on the left-hand side. Through mathematical manipulation, as shown in Equation (12), $u(t)$ can be represented as the product of a vector of knowns $[\phi_d]$ and a vector of unknowns $\{\hat{\theta}\}$. The vector of knowns corresponds to the desired trajectory, while the vector of unknowns represents the learned parameters obtained from proportional plus derivative feedback used to generate the process input. Therefore, the regression form of the process input u(t) is expressed as $u^*(t)$ in Equations (13) and (14).

$$u(t) = \frac{1}{0.0517}y(t+1) - \frac{1.7047}{0.0517}y(t) + \frac{0.7047}{0.0517}y(t-1) - u(t-1) \tag{12}$$

$$u^*(t) = \hat{a}_1 y_d(t+1) - \hat{a}_2 y_d(t) + \hat{a}_3 y_d(t-1) - \hat{b}_1 u_d(t-1) \tag{13}$$

$$u^*(t) = [\phi_d]\{\hat{\theta}\} = [y_d(t+1) - y_d(t) + y_d(t-1) - u_d(t-1)]\begin{Bmatrix}\hat{a}_1\\ \hat{a}_2\\ \hat{a}_3\\ \hat{b}_1\end{Bmatrix} \tag{14}$$

By utilizing feedforward control and transforming the state form $y(t)$ to $y(t+1)$ in Equation (12), it becomes possible to calculate the desired trajectory. The feedback parameters $[\hat{\theta}]$ are updated through a recursive least squares process, which requires initial rough estimates of the feedback parameters, the output y, and the regression $u^*(t)$, all described in reference [53]. Additionally, Equation (1)'s transfer function is converted back into an ordinary differential equation that is reparametrized as shown in Equation (13).

Conversely, in continuous deterministic artificial intelligence, converting the transfer function in Equation (1) to an ordinary differential equation (ODE) and re-parametrizing it based on Equation (13) is necessary. Another approach is to apply Equation (14) and utilize optimal feedback adjustment for learning feedback parameters in a discrete manure, as proposed by Smeresky [58].

$$u \equiv \phi_d(\phi_d^T\phi_d)^{-1}\phi_d^T\delta u \tag{15}$$

Table 2. Table of proximal variables and nomenclature [1].

Variable/acronym	Definition	Variable/acronym	Definition
u^*	Control input	Φ_d	Regressor matrix
y_d	Desired output	$\hat{\theta}$	Parameter vector
$\hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{b}_1$	Estimates	a_1, a_2, a_3, b_1	True values

[1] Such tables are offered throughout the manuscript to aid readability.

2.4. Autonomous Trajectory Generation

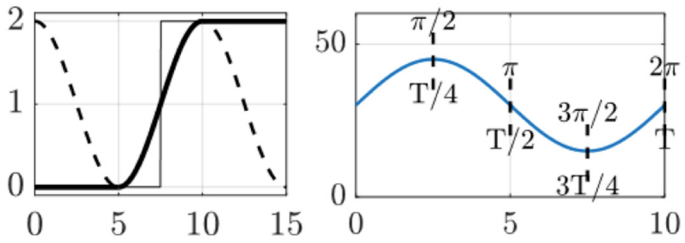
The purpose of autonomous trajectory generation for the DC motor or any other autonomous system is to generate the entire maneuver trajectory independently, based on a desired end state, without any external human assistance. While instantaneous maneuvering is possible in theory, the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

resulting trajectory would be a step function, lacking smoothness and containing discontinuities, making it impossible to differentiate. Therefore, an ideal trajectory should consist of three parts: the initial state, a smooth, continuously differentiable curve, and the desired end state.

2.4.1. Sinusoidal Trajectories

In most cases, system dynamics can be simplified or approximated to a basic ordinary differential equation of the form $\dot{z} = Az$, which can be solved using $z = Aexp(\lambda t)$. This solution can then be transformed into a sinusoidal function of the form $z = Asin(\omega t)$, as depicted in Figure 6. Therefore, one method of achieving autonomous trajectory generation is by computing a sinusoidal function based on the desired maneuver. The advantage of using a sinusoidal structure is that it can be easily differentiated and requires less computation compared to Pontryagin's method.



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

Figure 6. (**a**) A slew manipulation together with an approximated sinusoidal trajectory. (**b**) An ordinary sinusoidal trajectory.

To achieve a smooth maneuver, the ideal derivative of the initial and terminal points of the generated trajectory should be zero. In Figure 6b, at time $t = \frac{3T}{4}$, a smooth ascent is observed, indicating a derivative of zero that gradually increases, which represents an ideal starting point. The duration of the maneuver, represented by the hyperparameter Δt, determines how long the entire slew will take. Equation (16) shows that to complete a full cycle from the valley (lowest value) to the peak (highest value) and result in a sinusoidal wave, the period required should be twice the slew time.

$$\omega = \frac{2\pi}{T} = \frac{2\pi}{2\Delta t} = \frac{\pi}{\Delta t} \tag{16}$$

To ensure that our sinusoidal function matches the desired initial and terminal states, it is necessary to calculate the phase shift and manipulate the amplitude. The resulting output can be represented by Equation (17), where A_0 and A_f are the original and target states, respectively.

$$z = (A_f - A_0)[1 + \sin(\omega t + \phi)] \tag{17}$$

By joining the initial and the final state of the DC motor. The final state of the Sinusoidal based trajectory can be got, as shown in Equation (18).

$$for \begin{cases} t < tstart \\ tstart < t < tfinal \\ t > tfinal \end{cases} \quad \begin{matrix} \theta = A_0 \\ \theta = (A_f - A_0)[1 + \sin(\omega t + \phi)] \\ \theta = A_f \end{matrix} \tag{18}$$

2.4.2. Pontryagin's based Trajectories

Pontryagin's method is a theory of optimal control that is a special case of the Euler-Lagrange equation in calculus. It can calculate the optimal control signal that enables a dynamic system to transition from one state to another while accounting for boundary conditions on the state or input control. This theory was proposed in 1956 by the Soviet mathematician Lev Pontryagin and his students.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

Considering the dynamic function of the DC motor described by Equation (1), we can transform it using Equation (19) and obtain its dynamic equation as shown in Equation (20). This transformation allows us to model the behavior of the motor and analyze its response to various input signals more effectively.

$$Bs^2 + Bs = u, \qquad \ddot{B} + \dot{B} = u \tag{19}$$

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} x + \begin{pmatrix} 0 \\ u \end{pmatrix} \tag{20}$$

The Pontryagin's based trajectory generation method is a comprehensive process that involves several steps. It all begins by formulating a target quadratic function that needs to be minimized, with boundary requirements. This quadratic function is represented in Equation (21).

Minimize $J[x(\cdot), u(\cdot)] = \frac{1}{2}\int_{t_0}^{t_f} u^2 dt$

Subject to

$$\begin{gathered} \ddot{\theta} + \dot{\theta} = u \\ t_0 = \alpha \\ t_f = \beta \\ \theta_0 = A_0 \\ \theta_f = A_f \\ \dot{\omega}_0 = 0 \\ \dot{\omega}_f = 0 \end{gathered} \tag{21}$$

The state of the DC motor is described by a two-dimensional vector [θ, ω], which represents motor's angle and its angular velocity. The initial and final timestamps, t_0 and t_f, are manually selected prior to the calculation. The initial and final states correspond to the motor angle and angular velocity, both of which are zero.

Pontryagin's principle consists of the following steps for solving a boundary value problem: (1) Formulate a Hamiltonian function based on the given cost function and dynamics; (2) Minimize the Hamiltonian with respect to the state variable; (3) Calculate the co-state variable by taking the derivative of the Hamiltonian with respect to the state variable; (4) Generate appropriate boundary conditions to solve the boundary value problem.

The Hamilton function, denoted as H, is a combination of the target quadratic function and its boundary requirements, as shown in Equation (22). F represents the Lagrangian, λ is the costate or the adjoint variable, f(x,u) represents the system dynamics, x is the state variable, and u is the control variable.

Hamilton Function $\mathrm{H} = \mathrm{F} + \lambda^T f(x,u)$

$$\begin{gathered} H = \frac{1}{2}u^2 + \lambda_1 \dot{x} + \lambda_1 \ddot{x} \\ H = \frac{1}{2}u^2 + \lambda_1 \dot{x} + \lambda_2(-\dot{x} + u) \end{gathered} \tag{22}$$

The Pontryagin principle asserts that the optimal solution to an optimal control problem can be obtained by setting the derivative of the Hamiltonian with respect to the control variable to zero.

$$\begin{gathered} \frac{dH}{du} = 2u + \lambda_2 = 0 \\ u = -\frac{\lambda_2}{2} \end{gathered} \tag{23}$$

Afterwards, the next step involves equating the derivative of the Hamiltonian with respect to the system's states to the negative derivative of the co-states.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

$$\frac{dH}{d\dot{x}} = \dot{\lambda_1} = 0 \rightrightarrows \lambda_1 = a$$
$$\frac{dH}{d\ddot{x}} = \dot{\lambda_2} = \lambda_1 - \lambda_2 = a - \lambda_2 \quad (24)$$

Thus, the boundary value problem is now transformed as

$$\ddot{\theta} + \dot{\theta} = u$$
$$t_0 = \alpha$$
$$t_f = \beta$$
$$\theta_0 = A_0$$
$$\theta_f = A_f$$
$$\dot{\omega}_0 = 0 \quad (25)$$
$$\dot{\omega}_f = 0$$
$$u = -\frac{\lambda_2}{2}$$
$$\lambda_1 = a$$
$$\dot{\lambda_2} = a - \lambda_2$$

Solving the differential equation yields the function for the input value u

$$\lambda_2 = -e^{-t-c_1} + a$$
$$u = \frac{1}{2}e^{-t-c_1} - \frac{1}{2}a \quad (26)$$

After that the optimal trajectory can be calculated by taking the advantage of Equation (18). The c_2, c_1, c and a are all unknown parameters which can be calculated by the boundary equation.

$$\omega = e^{-t}(c_1 e^t + \frac{t}{2}e^{-c} + \frac{at}{2}e^t) - e^{-t}(\frac{1}{2}e^{-c} + c_1 e^t + \frac{a}{2}e^t + \frac{at}{2}e^t) - c_2 e^{-t}$$
$$\theta = c_2 e^{-t} - e^{-t}(c_1 e^t + \frac{t}{2}e^{-c} + \frac{at}{2}e^t) \quad (27)$$

Finally, an optimal trajectory based on Pontryagin's principle was constructed using a piecewise continuous function. The trajectory consisted of a period of inactivity, followed by the exponential function generated in the previous steps, which lasted for a duration of Δt. The trajectory then concluded with a period of constant final attitude.

$$for \begin{cases} t < tstart \\ tstart < t < tfinal \\ t > tfinal \end{cases} \quad \begin{matrix} \theta = A_0 \\ \theta = c_2 e^{-t} - e^{-t}(c_1 e^t + \frac{t}{2}e^{-c} + \frac{at}{2}e^t) \\ \theta = A_f \end{matrix} \quad (28)$$

3. Results

The purpose of this section is to provide a comprehensive comparison of several widely used model-following self-tuning techniques, including recursive least square, recursive least square with exponential forgetting, Auto Regressive Moving Average, extended least square, and extended least square with Posterior Residuals, when applied to DC-motors. The focus is on their performance. In this regard, after identifying the state-of-the-art methods in the previous section, it will then be compared with the discrete deterministic artificial intelligence approach presented in Equation (14). This comparison will be carried out using both sinusoidal and Pontryagin's based trajectory generation methods. Moreover, the comparison between continuous and discrete deterministic artificial intelligence will be analyzed in terms of trajectory following accuracy and input value range. It is expected that the results of this comparison will demonstrate the superiority of the discrete

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

deterministic artificial intelligence method augmented with Pontryagin's based trajectory generation method.

3.1. Different Model-following Self-tuner Control Benchmark

As explained in Section 2 of this paper, the model-following self-tuner is an estimation approach based on the model, which is used to generate control input to follow a desired trajectory. To implement Equation (14), various estimation techniques are used to adaptively calculate the unknown parameters. This section presents a comparison of the methods employed for an assumed model, followed by a comparison with the DC-motor model to identify the state-of-the-art methods for comparison with deterministic artificial intelligence.

$$y(t) = -\frac{9}{20}y(t-1) + \frac{3}{40}y(t-2) + \frac{1}{40}y(t-3) + 3u(t-1) + \frac{2}{3}u(t-2) - \frac{1}{9}u(t-3)+) - \frac{1}{10}\varepsilon(t) \quad (28)$$

Assuming a truth model with an uncorrelated output noise term ε(t)=N (0,1), history data for 1000-time samples are generated according to Equation (28). Figure 7 displays the estimation changes as the time stamp increases, while Table 3 provides detailed values regarding the differences between each method.

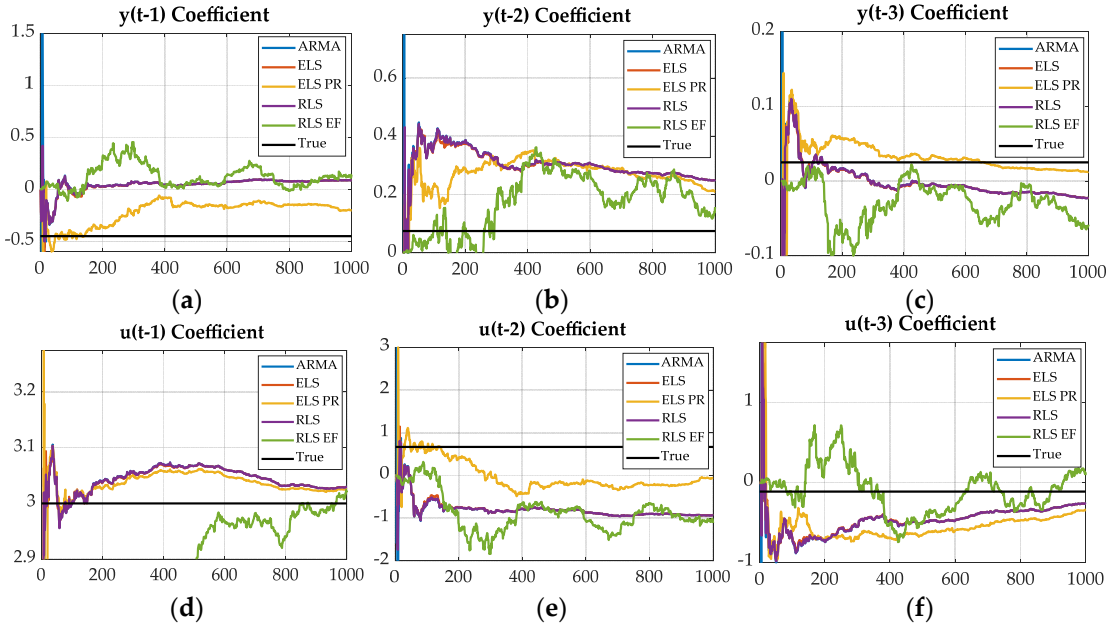


Figure 7. The result of different parameter estimation methods of an assumed truth model with respect to time on each respective abscissa, where ordinants display parameter estimates. (**a**) the $y(t-1)$ coefficient, (**b**) the $y(t-2)$ coefficient, (**c**) the $y(t-3)$ coefficient, (**d**) the $u(t-1)$ coefficient, (**e**) the $u(t-2)$ coefficient, (**f**) the $u(t-3)$ coefficient.

Figure 7 and table data indicate that extended least square with Posterior Residuals provides the most accurate estimates for the parameter y(t-1) and u(t-1), recursive least square with exponential forgetting is best for estimating y(t-2) and u(t-2), recursive least square yields the most accurate estimates for y(t-3), and Auto Regressive Moving Average is most effective for estimating u(t-3). This suggests that different methods have varying levels of efficacy for estimating different parameters. Determining the optimal method for a given task requires a comprehensive understanding of the employed model and the specific parameters being estimated. Consequently, the subsequent phase of this study involves implementing these methods on the DC motor to evaluate and compare their respective performances.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

Table 3. The performance difference between different estimation methods.

Parameter	True Value	ARMA	ELS	ELS PR	RLS	RLS EF
y(t-1)	-0.45	0.0788	0.0746	-0.2904	0.0786	0.0545
y(t-2)	0.075	0.1953	0.1926	0.2481	0.1952	0.3280
y(t-3)	0.025	-0.0265	-0.0264	0.0478	0.0265	0.0020
u(t-1)	3	2.9658	2.9663	2.9741	2.9658	3.0333
u(t-2)	0.667	-0.9241	-0.9111	0.1562	0.9235	0.8295
u(t-3)	-0.111	-0.1108	-0.1054	-0.5409	0.1108	0.5473

To achieve effective self-tuning of a system, it is crucial to design a control system that can convert unstable system responses into stable ones. This ensures that the output does not diverge and that the inputs can be tracked asymptotically. The step size of the system discretization is a critical factor that significantly impacts the overall tracking performance of the system. To evaluate the effectiveness of different parameter estimation methods for the DC motor system, we utilize three evaluation metrics: absolute mean error, absolute standard deviation, and total input value. These metrics consider both accuracy and system efficiency, enabling objective comparison among the estimation methods.

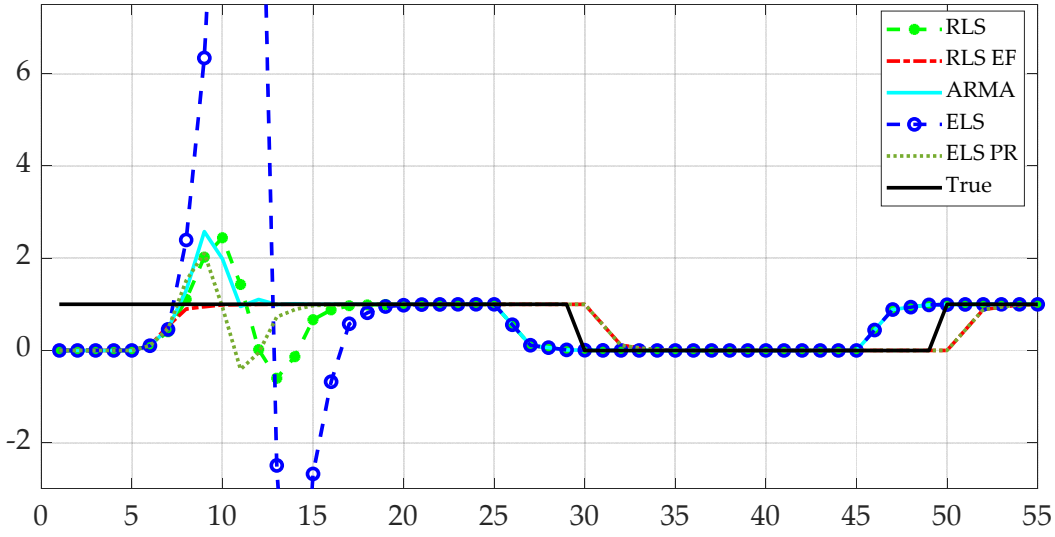


Figure 8. Comparation of different model following methods for DC motors: Input tracking of DC motors with model-following and parameter estimation with a step size t=0.35 with output y(t) on the ordinant versus time (seconds) on the abscissa.

The evaluation results presented in Table 3 demonstrate that the recursive least square with exponential forgetting method outperformed the other methods in terms of both mean error and error standard deviation, for step sizes of t = 0.35 and t = 0.5. The extended least square with Posterior Residuals method ranked second in performance, with an increase in mean error of 41.9% and 31.6%, and an increase in error standard deviation of 42.0% and 25.4%, respectively. Although the total input value evaluation indicated that the recursive least square with exponential forgetting method did not perform as well as the extended least square method, this factor is of relatively lower priority. Therefore, based on the comprehensive evaluation metrics, the recursive least square with exponential forgetting method is considered as the state-of-the-art model-following approach and will be used as the benchmark for evaluating deterministic artificial intelligence with different trajectory generation methods.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

Table 3. Error distribution of different Model Following methods with different step sizes.

Methods	Step Size	Error Mean	Error Standard Deviation	Mean Input Value
RLS	0.35	0.2106	0.4401	1.1083
RLS EF	0.35	0.1088	0.2960	1.2042
ARMA	0.35	0.1895	0.4142	4.7312
ELS	0.35	0.5192	2.0539	0.7548
ELS PR	0.35	0.1304	0.3323	0.7920
RLS	0.5	0.1971	0.4174	0.5073
RLS EF	0.5	0.1063	0.2938	0.6853
ARMA	0.5	0.2118	0.5036	1.4553
ELS	0.5	0.3621	1.1967	0.9825
ELS PR	0.5	0.1399	0.3686	0.3731

3.2. Discrete deterministic artificial intelligence with different trajectory generation methods

This section aims to apply the discrete deterministic artificial intelligence approach to the DC motor system with two different trajectory generation methods, namely, sinusoidal and Pontryagin's methods. The purpose is to compare the results obtained from these methods with the state-of-the-art recursive least square with exponential forgetting method discussed in Section 3.1. The evaluation of performance will be based on the mean error, standard deviation, and mean input value, which is consistent with the evaluation metrics used in the previous sections.

Upon examining the data presented in Table 4, it becomes apparent that the recursive least square with exponential forgetting method yields superior results compared to the deterministic artificial intelligence technique when the step size is set to 0.5. Nevertheless, it is crucial to recognize that a step size of this magnitude may not be suitable for real-world scenarios, as it could fail to capture the slew transfer accurately and timely. Consequently, it is the outcomes obtained using smaller step sizes that bear greater significance and practicality for implementation purposes. Therefore, it is vital to consider the performance of the methods under smaller step sizes when evaluating and selecting a model-following approach.

Table 4. Error distribution of DAI intelligence and RLSwEF with different step sizes.

Methods	Step Size	Error Mean	Error Standard Deviation	Input Value Mean
DAI (Sinusoidal)	0.35	0.0174	0.0573	0.2395
DAI (Pontryagin)	0.35	0.0128	0.0493	0.1771
RLS EF (Sinusoidal)	0.35	0.0185	0.0570	0.1756
RLS EF (Pontryagin)	0.35	0.0173	0.0551	0.1473
DAI (Sinusoidal)	0.4	0.0211	0.0571	0.3559
DAI (Pontryagin)	0.4	0.0154	0.0457	0.2346
RLS EF (Sinusoidal)	0.4	0.0211	0.0639	0.1588
RLS EF (Pontryagin)	0.4	0.0197	0.0612	0.1447
DAI (Sinusoidal)	0.5	0.0711	0.1175	0.8794
DAI (Pontryagin)	0.5	0.0612	0.1104	0.7861
RLS EF (Sinusoidal)	0.5	0.0261	0.0807	0.1414
RLS EF (Pontryagin)	0.5	0.0242	0.0805	0.1365

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

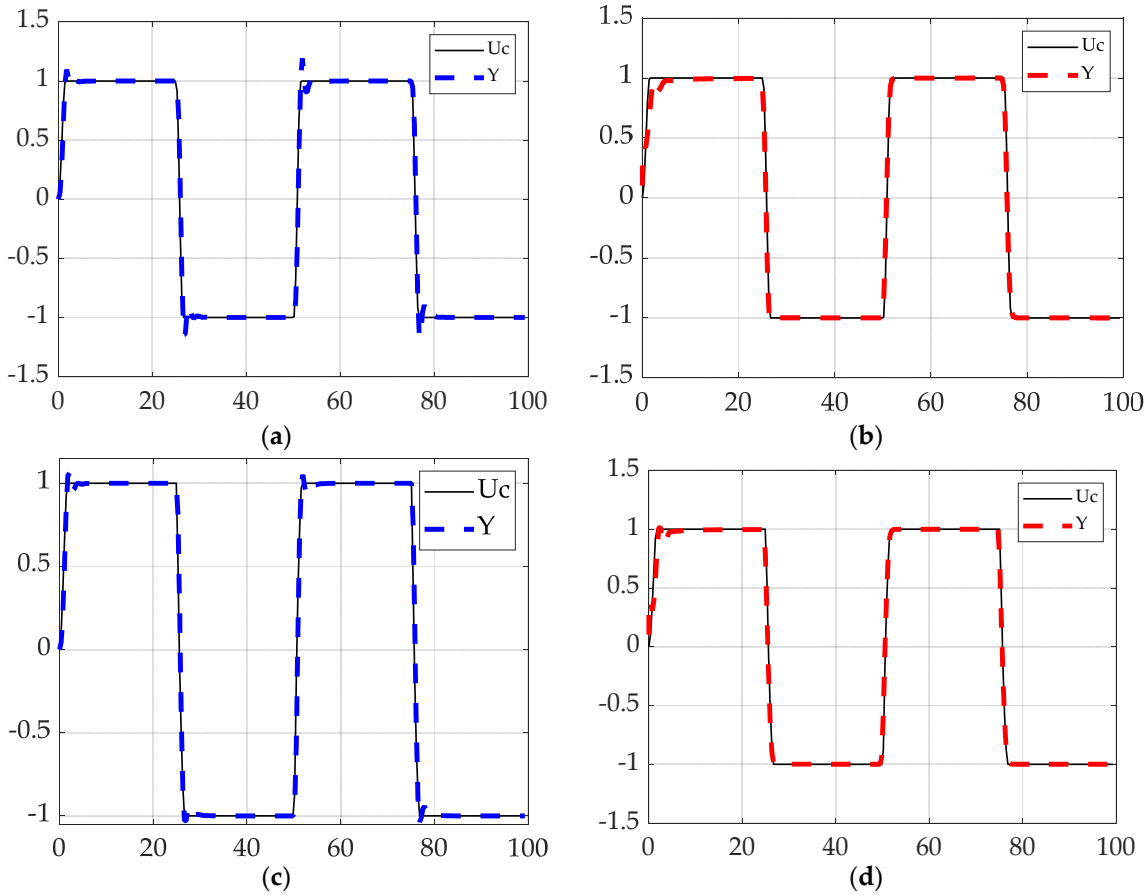


Figure 9. The output signal for the control approaches with a 0.35 second step size. The black line represents the command signal. (**a**) Output signal obtained from deterministic artificial intelligence with Sinusoidal trajectory Discrete DAI using Sinusoidal; (**b**) Output signal obtained from RLS EF based MF method with Sinusoidal trajectory; (**c**) Output signal obtained from deterministic artificial intelligence with Pontryagin based trajectory; (**d**) Output signal obtained from RLS EF based MF method with Pontryagin based trajectory.

As the time step decreases, the performance of the evaluated methods becomes more reliable. Specifically, for a time step of 0.4, both the deterministic artificial intelligence and recursive least square with exponential forgetting methods exhibit the same mean error. However, the deterministic artificial intelligence approach shows a slightly smaller standard deviation but a larger input value. This suggests that while both methods have similar accuracy, the deterministic artificial intelligence approach may be more efficient in terms of resource usage. However, upon decreasing the time step to 0.35, the deterministic artificial intelligence approach outperforms the recursive least square with exponential forgetting method in both mean error and standard deviation, thereby validating the effectiveness of the former approach. It is worth noting that the output of deterministic artificial intelligence exhibits an overshoot at discontinuities and tracks the input signal with a small tracking error, consistent with the observations made by Koo [9]. Moreover, it is worth highlighting that the increase in output observed upon decreasing the step size from 0.4 to 0.35 is not indicative of a decrease in performance but rather reflects the fact that more time steps are required to output larger values.

After conducting our analysis, we can also conclude that the Pontryagin-based trajectory generation method consistently outperforms the sinusoidal method across three evaluation metrics when the same method and step size are employed. This finding provides strong evidence of the effectiveness of the Pontryagin's method. Moreover, this observation is consistent with the underlying mathematical principle, as the Pontryagin's method seeks an optimal trajectory directly from the functional, resulting in a non-approximate solution. In contrast, the sinusoidal trajectory is an approximate solution, and therefore, it is expected to be less effective.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

3.3. Continous deterministic artificial intelligence with different trajectory generation methods

Compared to discrete deterministic artificial intelligence, continuous deterministic artificial intelligence utilizes ordinary differential equations to simulate system dynamics. The table below presents a comparative analysis between the two approaches, considering both sinusoidal and Pontryagin's based methodologies.

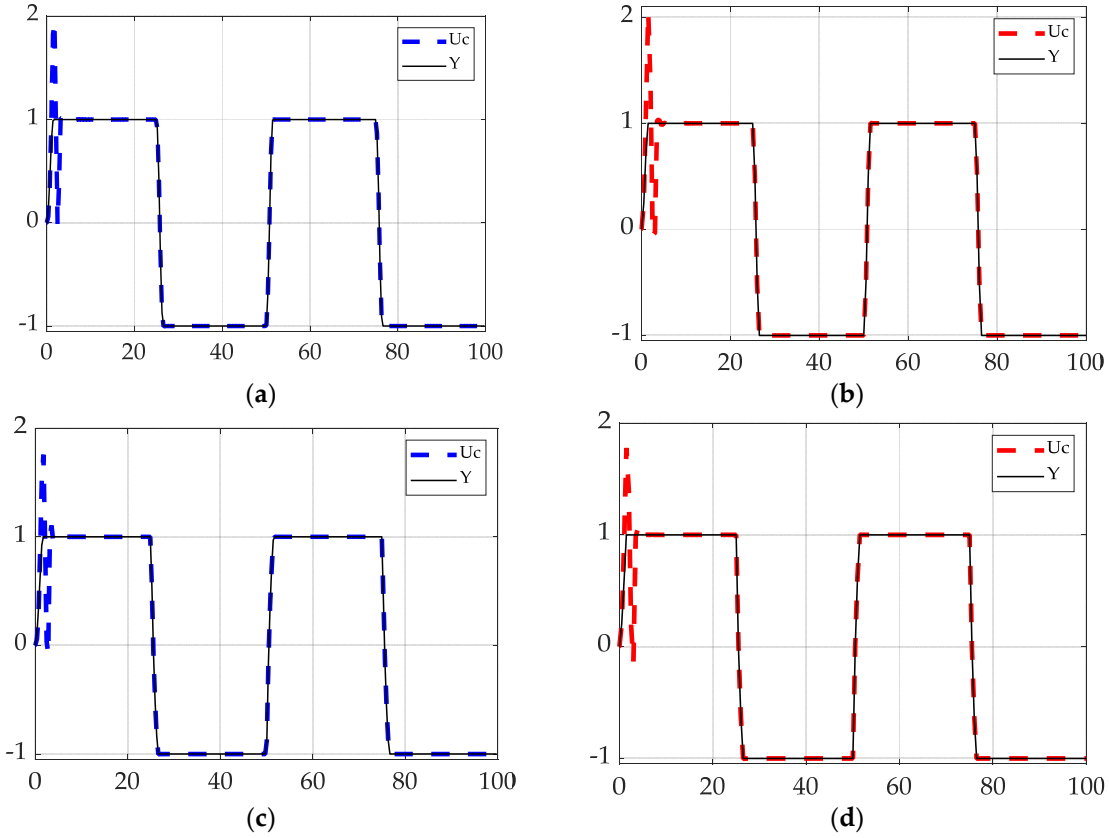


Figure 10. The black line represents the command signal. (**a**) Output signal obtained from Continuous DAI with Sinusoidal trajectory with step size=0.35s; (**b**) Output signal obtained from Continuous DAI with Sinusoidal trajectory with step size=0.5s; (**c**) Output signal obtained from Continuous DAI with Pontryagin's based trajectory with step size=0.35s; (**d**) Output signal obtained from Continuous DAI with Pontryagin's based trajectory with step size=0.5s.

Table 5 demonstrates that the continuous deterministic artificial intelligence outperforms the discrete alternative with a time step size of 0.5, but the discrete deterministic artificial intelligence with the Pontryagin-based methodology achieves superior performance when the time step size decreases to 0.35. The robustness of continuous deterministic artificial intelligence across varying time step sizes further confirms its reliability. Moreover, Pontryagin's method consistently outperforms the sinusoidal method in terms of mean error and standard deviation, indicating its effectiveness. However, it is worth noting that sometimes Pontryagin's method may require larger input values than the sinusoidal method, potentially due to errors in solving ordinary differential equations. This issue could be explored further in future work.

Table 5. Error Distribution of comparison between discrete deterministic artificial intelligence and Continuous deterministic artificial intelligence with different step sizes.

DAI Type	Step Size	Error Mean	Error Standard Deviation	Input Value Mean
Discrete (Sinusoidal)	0.35	0.0174	0.0573	0.2395
Discrete (Pontryagin)	0.35	0.0128	0.0493	0.1771
Continuous (Sinusoidal)	0.35	0.0141	0.1024	2.3332

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

Continuous (Pontryagin)	0.35	0.0137	0.1026	14.3891
Discrete (Sinusoidal)	0.5	0.0711	0.1175	0.8794
Discrete (Pontryagin)	0.5	0.0612	0.1104	0.7861
Continuous (Sinusoidal)	0.5	0.0203	0.1320	4.3319
Continuous (Pontryagin)	0.5	0.0177	0.1166	5.9983

4. Discussion

The table presented below demonstrates the performance improvement achieved by different algorithms when utilizing different trajectory generation methods. Our findings validate the effectiveness of the deterministic artificial intelligence approach, particularly when combined with the Pontryagin-based trajectory generation method, in accurately tracking discontinuous command square waves compared to alternative techniques. Also, it should be noted that continuous deterministic artificial intelligence requires large input values, which may represent a potential limitation when applying this approach to real-world problems.

Table 6. Performance difference for deterministic artificial intelligence and MF benchmark with different step sizes.

Methods	Step Size	Error Mean	Error Standard Deviation	Input Value Mean
DAI (Sinusoidal)	0.35	-17.5%	0.35%	-32.70%
DAI (Pontryagin)	0.35	-39.33%	-13.66%	-50.23%
RLS EF(Sinusoidal)	0.35	-12.32%	-0.17%	-50.66%
RLS EF(Pontryagin)	0.35	-18.00%	-3.50%	-58.61%
DAI (Sinusoidal)	0.4	0%	0%	0%
DAI (Pontryagin)	0.4	-27.0%	-19.96%	-34.08%
RLS EF(Sinusoidal)	0.4	0%	7.18%	-55.38%
RLS EF(Pontryagin)	0.4	-6.63%	7.18%	-59.34%
DAI (Sinusoidal)	0.5	236.96%	105.77%	147.09%
DAI (Pontryagin)	0.5	190.04%	93.34%	120.87%
RLS EF(Sinusoidal)	0.5	23.69%	41.33%	-60.26%
RLS EF(Pontryagin)	0.5	14.69%	40.98%	-61.64%

Table 7. Performance difference for discrete and continuous deterministic artificial intelligence with different step sizes.

DAI Type	Step Size	Error Mean	Error Standard Deviation	Input Value Mean
Discrete (Sinusoidal)	0.35	-14.28%	-56.59%	-94.47%
Discrete (Pontryagin)	0.35	-36.94%	-62.65%	-95.91%
Continuous (Sinusoidal)	0.35	-30.54%	-22.42%	-46.13%
Continuous (Pontryagin)	0.35	-32.51%	-22.27%	232.16%
Discrete (Sinusoidal)	0.5	250.24%	-10.98%	-79.64%
Discrete (Pontryagin)	0.5	201.47%	-16.36%	-81.85%
Continuous (Sinusoidal)	0.5	0%	0%	0%
Continuous (Pontryagin)	0.5	-12.8%	0.16.21%	38.46%

Our manuscript presents a comprehensive analysis of the control effects of various control algorithms and compares their performance at different step sizes. We observe that as the step size decreases, the discrepancy between the output signals generated by different algorithms gradually decreases, eventually becoming negligible. Additionally, our findings indicate that the Pontryagin-based trajectory generation method consistently outperforms the sinusoidal method due to its superior mathematical properties. In terms of overall performance, discrete deterministic artificial intelligence appears to be the most effective at small step sizes, followed by continuous deterministic artificial intelligence. However, since continuous deterministic artificial intelligence offers superior

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1

performance across various computation rates compared to other approaches, it may represent the most feasible and reliable option for future applications on unmanned vehicles, provided its input value does not exceed the input limit.

4.1. Conclusion

Our manuscript presents a simulation-based comparison of the performance of continuous and discrete deterministic artificial intelligence models, along with a Model-Following benchmark, using both Pontryagin's and Sinusoidal trajectory generation methods on the DC-motor. Our findings reveal the superiority of discrete deterministic artificial intelligence augmented with Pontryagin's method, particularly at small step sizes.

However, although continuous deterministic artificial intelligence demonstrates robustness across different time steps, our study also highlights a potential limitation in real-world applications. Specifically, we find that it may require large input values that could exceed the maximum input capacity of the DC motor. This observation suggests that careful consideration must be given to input values when using continuous deterministic artificial intelligence models for practical applications.

Overall, our study provides valuable insights into the performance of different artificial intelligence models and trajectory generation methods in the context of a DC motor. Our results can inform the development of more effective and practical control systems for similar unmanned vehicle applications.

4.2. Future Research

This paper presents several novel discoveries, but there are also several areas for future exploration. One such area is the need for further research into the reason for the larger input values required when combining continuous deterministic artificial intelligence with Pontryagin's method. Additionally, developing an effective method to eliminate overshot and errors in the initial part of continuous deterministic artificial intelligence and the transient part of discrete deterministic artificial intelligence would enhance their appeal for real-world applications.

Author Contributions: Conceptualization, J.X. and T.S.; methodology, J.X. and T.S.; software, J.X. and T.S.; validation, T.S.; formal analysis, J.X. and T.S.; investigation, J.X. and T.S.; resources, T.S; writing - original draft preparation, J.X.; writing - review and editing, J.X. 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 obtained by contacting the corresponding author.

Acknowledgments: The code used to generate data and figures in this manuscript is adapted from the Appendix of Rohan Shah [8], Koo Mo Koo, and Henry Travis's article [9]

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

Appendix A

To understand and replicate the research presented in this manuscript, it is crucial to refer to the appendix. MATLAB® R2021b is the required software for running the code and other MATLAB® versions may cause unexpected error.

Appendix A.1. Model-Following Benchmarks for DC Motor

```
clear all;clc;
rand('seed',1);
Bp=[0 0 1];Ap=[1 1 0];Gs=tf(Bp,Ap); %Create continuous time transfer function
Ts=0.35; Hd=c2d(Gs,Ts,'matched'); % Transform continuous system to discrete system
B = Hd.Numerator{1}; A = Hd.Denominator{1};
a1=0;a2=0;b0=0.1;b1=0.2;
```

 doi:10.20944/preprints202304.0010.v1


```
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 = 1000000;
%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
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))
            check = 1;
            run_next = 0;
        end
    end
end

%%%%%%%%%%%%%%%%RECURSIVE LEAST
SQUARES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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)]';
    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
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
    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%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%AUTOREGRESSIVE MOVING
AVERAGE%%%%%%%%%%%%%%%%%%%%
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];
    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];
    if (i > 3)
        THETA_hat(:,i+1) = inv(phi'*phi)*phi'*Y(1+nzeros:t)';
    else
        THETA_hat(:,i+1) = THETA_hat(:,i);
    end
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
    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%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%EXTENDED LEAST SQUARES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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=[ones(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];
    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;
    epsilon(t)=Y(t)-phi'*theta_hat_els(:,i);
    theta_hat_els(:,i+1)=theta_hat_els(:,i)+K*epsilon(t);
    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;
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
    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%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%DETERMINISTIC AI%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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);
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;
    err = hvy_m(t)-Y(t);
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
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%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%RECURSIVE LEAST SQUARES w/exponential forgetting%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Uc = Uc(1:200);
n=4;lambda=0.99;
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)]';
    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 = [plotu; U];
ploty = [ploty; Y(1:205)];
plottheta = [plottheta; THETA_hat];
%%%%%%%%%%%%%%%%%%%%%%%%%END OF RECURSIVE LEAST SQUARES w/exponential forgetting%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%EXTENDED LEAST SQUARES With Posterior Residuals %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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];
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
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];
    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 With Posterior
Residuals %%%%%%%%%%%%%%%%%%%%

y_rls = ploty(1,:);
y_arma = ploty(2,:);
y_els = ploty(3,:);
y_dai = ploty(4,:);
y_rlswef = ploty(5,:);
y_elswpr = ploty(6,:);
plot(y_rls,'g--*','LineWidth',2)
hold on
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
plot(y_rlswef,'r-.','LineWidth',2)
hold on
plot(y_arma,'c','LineWidth',2)
hold on
plot(y_els,'b--o','LineWidth',2)
hold on
plot(y_elswpr,':','LineWidth',2)
hold on
plot(Uc,'k-','LineWidth',2)
legend('RLS','RLSwEF','ARMA','ELS','ELSwPR','True','fontsize',11);
set(gca,'fontname','Palatino Linotype');
grid on
axis([0 55,-1.5,2.5]);
title('Comparation of different Model Following Method for DC Motor')
xlabel('Time step (in sec)'); ylabel('Output (Y)');

mean_abs_rls = mean(abs(y_rls - Uc))
mean_abs_rlswef = mean(abs(y_rlswef - Uc))
mean_abs_arma = mean(abs(y_arma - Uc))
mean_abs_els = mean(abs(y_els - Uc))
mean_abs_elswpr = mean(abs(y_elswpr - Uc))
std_abs_rls = std((y_rls - Uc))
std_abs_rlswef = std((y_rlswef - Uc))
std_abs_arma = std(y_arma - Uc)
std_abs_els = std((y_els - Uc))
std_abs_elswpr = std((y_elswpr - Uc))
mean_rls_input = mean(abs(plotu(1,:)))
mean_rlswef_input = mean(abs(plotu(2,:)))
mean_arma_input = mean(abs(plotu(3,:)))
mean_els_input = mean(abs(plotu(4,:)))
mean_elswpr_input = mean(abs(plotu(5,:)))
```

Appendix A.2 Discrete deterministic artificial intelligence and MF benchmark with Pontryagin's method

```
clear all; clc; close all;
rand('seed',1);
%% DISCRETIZATION
% B=[0 0.1065 0.0902];A=poly([1.1 0.8]);
% Gs = tf(B,A);
% a1=0;a2=0;b0=0.1;b1=0.2; %Shah's
Bp=[0 0 1];Ap=[1 1 0];Gs=tf(Bp,Ap); %Create continuous time transfer function
Ts=0.5; Hd=c2d(Gs,Ts,'matched'); % Transform continuous system to discrete system
B = Hd.Numerator{1}; A = Hd.Denominator{1};
b0=0.1; b1=0.1; a0=0.1; a1=0.01; a2=0.01;

%% RLSwEF
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;
Rmat=[];
factor = 25;
% Reference
T_ref = 25; t_max = 100; time = 0:Ts:t_max; nt = length(time);
% slew stuff
Tslew = 1.5; Uc = zeros(length(nt));
syms C2 t C1 c a
y(t) =C2*exp(-t) - exp(-t)*(C1*exp(t) + (t*exp(-c))/2 + (a*t*exp(t))/2);
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
ydot(t) = exp(-t)*(C1*exp(t) + (t*exp(-c))/2 + (a*t*exp(t))/2) - exp(-t)*(exp(-c)/2 + C1*exp(t) + (a*exp(t))/2 + (a*t*exp(t))/2) - C2*exp(-t);
for j=1:nt
    % pos or neg
    if mod(time(j),2*T_ref)<T_ref
        pn = 1;
    else
        pn =-1;
    end
    % slew
    if mod(time(j),T_ref)<Tslew

        if time(j)>=0 && time(j)<Tslew
            eqns = [y(0) == 0,ydot(0) == 0 ,y(Tslew) == 1,ydot(0) == 0];
            S = solve(eqns,[C2 C1 c a]);
            Uc(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 + (S.a.*time(j).*exp(time(j)))/2);
        else if time(j)>=50 && time(j)<50 + Tslew
                t = time(j);
                eqns = [y(50) == -1,ydot(0) == 0 ,y(50 + Tslew) == 1,ydot(0) == 0];
                S = solve(eqns,[C2 C1 c a]);

                Uc(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 + (S.a.*time(j).*exp(time(j)))/2);
        else if time(j)>=25 && time(j)<25 + Tslew
                t = time(j);
                eqns = [y(25) == 1,ydot(0) == 0 ,y(25 + Tslew) == -1,ydot(0) == 0];
                S = solve(eqns,[C2 C1 c a]);
                Uc(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 + (S.a.*time(j).*exp(time(j)))/2);
        else if time(j)>=75 && time(j)<75 + Tslew
                t = time(j);
                eqns = [y(75) == 1,ydot(0) == 0 ,y(75 + Tslew) == -1,ydot(0) == 0];
                S = solve(eqns,[C2 C1 c a]);
                Uc(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 + (S.a.*time(j).*exp(time(j)))/2);
        else if time(j)>=100 && time(j)<100 + Tslew
                t = time(j);
                eqns = [y(100) == -1,ydot(0) == 0 ,y(100 + Tslew) == 1,ydot(0) == 0];
                S = solve(eqns,[C2 C1 c a]);
                Uc(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 + (S.a.*time(j).*exp(time(j)))/2);
        end
        end
        end
        end
        end
    else
        Uc(j)=pn;
    end
end
n=4;lambda=0.95;
nzeros=2;time=zeros(1,nzeros);Y=zeros(1,nzeros);Ym=zeros(1,nzeros);
U=ones(1,nzeros);Uc=[zeros(1,nzeros),Uc];
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
Noise = 0;
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:nt
    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)]';
    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];
    %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];

    Rmat=[Rmat 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
U_rlswef = U;

%% DAI
%Create command signal, Uc based on Example 3.5 plots...square wave with 50 sec period
t_max = nt-1;
THETA_hat(:,1)=[-a1 -a2 b0 b1]';
n = length(THETA_hat);
% Sigma=1/25; Noise=Sigma*randn(nt,1);
% Noise = 0;
nzeros=2;
Y_true=zeros(1,nzeros);Ym=zeros(1,nzeros);U=zeros(1,nzeros);
P=[100 0 0 0;0 100 0 0;0 0 1 0;0 0 0 1];
lambda = 1;
eb = Y_true(1) - Uc(1);
err = 0;
kp = 2.0;
kd = 6.0;
hatvec = zeros(4,1);
for i=1:t_max+1 %Loop through the output data Y(t)
    t=i+nzeros;
    de = err-eb;
    u = kp*err + kd*de;
    U(t-1) = u;
    Y_true(t)=[Y_true(t-1) Y_true(t-2) U(t-1) U(t-2)]*[-A(2) -A(3) B(2) B(3)]';
    phid = [Y_true(t) -Y_true(t-1) Y_true(t-2) -U(t-2)];
    newest = phid\u;
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
    hatvec(:,i) = newest;
    eb = err;
    %disp(t);
    err = Uc(t)-Y_true(t);
end
U_DAI = U;

%% PLOT
tspan = linspace(0,100,nt);
tspan = [zeros(1,2) tspan];
figure(1); %DAI
plot(tspan(1:nt),Uc(1:nt),'k-','LineWidth',1); hold on; plot(tspan(1:nt),Y_true(2:nt+1),'b--','LineWidth',3); hold off
xlabel('Time(sec)');ylabel('Output(Y)'); title('Discrete DAI using Pontryagin T = 0.35');
legend('Uc','Y','fontsize',11);
set(gca,'fontsize',16); set(gca,'fontname','Palatino Linotype'); xlim([0 max(time)]); grid;
% p=plot(tspan,Uc(1:203),'-',tspan,Y,'-'); p(2).LineWidth = 2; legend('Uc','Y','fontsize',11); %DAI
axis([0 100,-1.5 1.5]);
figure(2); %RLS estimation
plot(tspan(1:nt),Uc(1:nt),'k-','LineWidth',1); hold on; plot(tspan(1:nt),Y(3:nt+2),'r--','LineWidth',3); hold off
xlabel('Time(sec)');ylabel('Output(Y)'); title('RLSwEF using Pontryagin T = 0.35'); legend('Uc','Y','fontsize',11);
set(gca,'fontsize',16); set(gca,'fontname','Palatino Linotype'); xlim([0 max(time)]); grid;
axis([0 100,-1.5 1.5]);
DAI_err_mean = mean(abs(Uc(1:nt)-Y_true(2:nt+1)))
DAI_err_std = std(abs(Uc(1:nt)-Y_true(2:nt+1)))
RLS_err_mean = mean(abs(Uc(1:nt)-Y(3:nt+2)))
RLS_err_std = std(abs(Uc(1:nt)-Y(3:nt+2)))
Uinput_sum_RLS = mean(abs(U_rlswef))
Uinput_sum_DAI = mean(abs(U_DAI))
Appendix A.3 Continuous deterministic artificial intelligence with Pontryagin's method
clear all;clc;close all;
rand('seed',1);
% Enter Given Plant parameters
for k=1:2
    Bp=[0 0 1];Ap=[1 1 0];Gs=tf(Bp,Ap); %Create continuous time transfer function
    Ts=[0.5 0.35]; Hz=c2d(Gs,Ts(k),'matched'); % Transform continuous system to discrete system
    B = Hz.Numerator{1}; A = Hz.Denominator{1};
    % Initial estimates of plant parameters for undetermined system from example 3.5
    b0=0.1; b1=0.1; a0=0.1; a1=0.01; a2=0.01;
    % Reference
    T_ref = 25; t_max = 100; time = 0:Ts(k):t_max; nt = length(time);
    % slew stuff
    syms C2 t C1 c a
    Tslew = 1.5; Yd = zeros(length(nt));
    y(t) =C2*exp(-t) - exp(-t)*(C1*exp(t) + (t*exp(-c))/2 + (a*t*exp(t))/2);
    ydot(t) = exp(-t)*(C1*exp(t) + (t*exp(-c))/2 + (a*t*exp(t))/2) - exp(-t)*(exp(-c)/2 + C1*exp(t) + (a*exp(t))/2 +
(a*t*exp(t))/2) - C2*exp(-t);
    for j=1:nt
        % pos or neg
        if mod(time(j),2*T_ref)<T_ref
            pn = 1;
        else
            pn =-1;
        end
        % slew
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2023 doi:10.20944/preprints202304.0010.v1


```
        if mod(time(j),T_ref)<Tslew
            if time(j)>=0 && time(j)<Tslew
                eqns = [y(0) == 0,ydot(0) == 0 ,y(Tslew) == 1,ydot(0) == 0];
                S = solve(eqns,[C2 C1 c a]);
                Yd(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 +
(S.a.*time(j).*exp(time(j)))/2);
            else if time(j)>=50 && time(j)<50 + Tslew
                    t = time(j);
                    eqns = [y(50) == -1,ydot(0) == 0 ,y(50 + Tslew) == 1,ydot(0) == 0];
                    S = solve(eqns,[C2 C1 c a]);

                    Yd(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 +
(S.a.*time(j).*exp(time(j)))/2);
            else if time(j)>=25 && time(j)<25 + Tslew
                    t = time(j);
                    eqns = [y(25) == 1,ydot(0) == 0 ,y(25 + Tslew) == -1,ydot(0) == 0];
                    S = solve(eqns,[C2 C1 c a]);
                    Yd(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 +
(S.a.*time(j).*exp(time(j)))/2);
            else if time(j)>=75 && time(j)<75 + Tslew
                    t = time(j);
                    eqns = [y(75) == 1,ydot(0) == 0 ,y(75 + Tslew) == -1,ydot(0) == 0];
                    S = solve(eqns,[C2 C1 c a]);
                    Yd(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 +
(S.a.*time(j).*exp(time(j)))/2);
            else if time(j)>=100 && time(j)<100 + Tslew
                    t = time(j);
                    eqns = [y(100) == -1,ydot(0) == 0 ,y(100 + Tslew) == 1,ydot(0) == 0];
                    S = solve(eqns,[C2 C1 c a]);
                    Yd(j)=S.C2.*exp(-time(j)) - exp(-time(j)).*(S.C1.*exp(time(j)) + (time(j).*exp(-S.c))/2 +
(S.a.*time(j).*exp(time(j)))/2);
            end
            end
            end
            end
            end
        else
            Yd(j)=pn;
        end
    end
    THETA_hat(:,1)=[-a1 -a2 b0 b1]';
    n = length(THETA_hat);
    Sigma=1/12*0; Noise=Sigma*randn(nt,1);
    nzeros=2;Y=zeros(1,nzeros);Y_true=zeros(1,nzeros);
    Ym=zeros(1,nzeros);U=zeros(1,nzeros);Yd=[zeros(1,nzeros),Yd];
    P=[100 0 0 0;0 100 0 0;0 0 1 0;0 0 0 1];
    lambda = 1;
    for i=1:nt-1
        t=i+nzeros;
        % Update Dynamics
        Y_true(t)=[Y(t-1) Y(t-2) U(t-1) U(t-2)]*[-A(2) -A(3) B(2) B(3)]';
        Y(t)=Y_true(t)+Noise(i);
        phi=[Y(t-1) Y(t-2) U(t-1) U(t-2)]';
        K=P*phi*1/(lambda+phi'*P*phi);
```

 doi:10.20944/preprints202304.0010.v1


```
        P=P-P*phi/(1+phi'*P*phi)*phi'*P/lambda;
        innov_err(i)=Y(t)-phi'*THETA_hat(:,i);
        THETA_hat(:,i+1)=THETA_hat(:,i)+K*innov_err(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);%
        THETA=[-a1 -a2 b0 b1];
        % Calculate Model control, U(t) optimally
        U(t)=[Yd(t+1) Y(t) Y(t-1) U(t-1)]*[1 a1 a2 -b0]'/b1;
    end
    Y_true(end+1)=Y_true(end);
    FS = 2;
    time = [-(nzeros-1)*Ts:Ts:0 time];
    Ts(k)
    DAI_err_mean = mean(abs(Yd-Y_true))
    DAI_err_std = std(abs(Yd-Y_true))
    U_input = mean(abs(U))
    if k==1
        figure (k)
        h1 = plot(time,Y_true,'r--','LineWidth',3);hold on;
        plot(time,Yd,'k-','LineWidth',1);
        axis([0 100,-1.5 1.5]); hold off; grid;
        legend('Uc','Y','fontsize',11);     xlabel('Time(sec)');ylabel('Output(Y)');title('Continuous     DAI     with
Pontryagin T = 0.50s'); set(gca,'fontsize',16);
        set(gca,'fontname','Palatino Linotype');
    else
        figure (k)
        h1 = plot(time,Y_true,'b--','LineWidth',3);hold on;
        plot(time,Yd,'k-','LineWidth',1);
        axis([0 100,-1.5 1.5]); hold off; grid;
        legend('Uc','Y','fontsize',11);     xlabel('Time(sec)');ylabel('Output(Y)');title('Continuous     DAI     with
Pontryagin T = 0.35s'); set(gca,'fontsize',16);
        set(gca,'fontname','Palatino Linotype');
    end
end
```

References

1. Blake, T. What is Unmanned Aircraft Systems Traffic Management? May 27, 2021. Available online: https://www.nasa.gov/ames/utm (accessed 3 March 2023).
2. Tabor, A. New Era Begins as Drone Traffic Management Project Wraps Up. Jun 22, 2021. Available online: https://www.nasa.gov/feature/ames/new-era-begins-as-drone-traffic-management-project-wraps-up (accessed 3 March 3, 2023).
3. NASA Image Use Policy. Available online: https://gpm.nasa.gov/image-use-policy (accessed on 3 March 2023).
4. Guerges, M. NASA Integrates Gear Motors For Robotic Arm on Future Lunar Missions. Available online: https://www.nasa.gov/feature/nasa-integrates-gear-motors-for-robotic-arm-on-future-lunar-missions (accessed on 19 February 2023).
5. Apkarian, J.; Åström, J. A laptop servo for control education. IEEE Contr. Syst. Mag., 2004 24(5), 70–73.
6. Bernstein, D. The Quanser DC Motor Control Trainer. IEEE Contr. Syst. Mag. 2005, 25(3), 90¬–93. Available online: http://www-personal.umich.edu/~dsbaero/others/35-QuanserDCMotor.pdf (accessed 15 February 2023).
7. Hoque, M.A., M. R. Zaman, and M. A. Rahman, Artificial neural network based permanent magnet dc motor drives. in Proc. IEEE-IAS Annu. Meeting, 1995, 1, 98–103.
8. Hoque, M., Zaman, M.; Rahman, M. Artificial neural network based controller for permanent magnet dc motor drives, in Proc. IEEE-IAS Annu. Meeting 1995, 2, 1775–1780.
9. Khomenko, M.; Voytenko, V.; Vagapov, Y. Neural network-based optimal control of a DC motor positioning system. *International Journal of Automation and Control* **2013** *7*(1-2), 83-104.

10. Yogesh; Gupta, S.; Garg, M. DC Motor Speed Control using Artificial Neural Network. *International Journal of Modern Communication Technologies & Research* **2014**, *2*(2), 19-24.
11. Naung, Y.; Anatolii, S.; Lin, Y. Speed Control of DC Motor by Using Neural Network Parameter Tuner for PI-controller. IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Saint Petersburg and Moscow, Russia, 2019, pp. 2152-2156.
12. Yang, X.; Deng, W.; Yao, J. Neural network based output feedback control for DC motors with asymptotic stability. *Mechanical Systems and Signal Processing* **2022**, *164*(1), 108288.
13. Ćirić, D.; Janković, M.; Miletić, M. Sound Based DC Motor Classification by a Convolution Neural Network. 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), Ohrid, North Macedonia, 2022, pp. 1-4, doi: 10.1109/ICEST55168.2022.9828682.
14. Nizami, T.; Gangula, S.; Reddy, R.; Dhiman, H. Legendre Neural Network based Intelligent Control of DC-DC Step Down Converter-PMDC Motor Combination. IFAC – PapersOnLine 2022, *55*(1), 162-167.
15. Lei, G.; Xie, L.; Ni, W.; Zheng, S. Self-tuning and approximation via RBF neural networks. CCEAI '23: Proceedings of the 7th International Conference on Control Engineering and Artificial Intelligence. January 2023 Pages 72–78https://doi.org/10.1145/3580219.3580233.
16. Scalabrini Sampaio, G.; Vallim Filho, A.R.d.A.; Santos da Silva, L.; Augusto da Silva, L. Prediction of Motor Failure Time Using An Artificial Neural Network. *Sensors* **2019**, *19*, 4342.
17. Zhang, L.; Wang, Y.; Cheng, L.; Kang, W. A Three-Parameter Adaptive Virtual DC Motor Control Strategy for a Dual Active Bridge DC–DC Converter. *Electronics* **2023**, *12*, 1412.
18. Tufenkci, S.; Alagoz, B.; Kavuran, G.; Yeroglu, C.; Herencsar, N.; Mahata, S. A theoretical demonstration for reinforcement learning of PI control dynamics for optimal speed control of DC motors by using Twin Delay Deep Deterministic Policy Gradient Algorithm. *Expert Systems with Applications* **2023**, *213*(C), 119192.
19. Munagala, V.K., Jatoth, R.K. A novel approach for controlling DC motor speed using NARXnet based FOPID controller. Evolving Systems 2023 14, 101–116.
20. Prakash, A.; Naveen, C. Combined strategy for tuning sensor-less brushless DC motor using SEPIC converter to reduce torque ripple. *ISA Transactions* **2023**, *133*, 328-344.
21. Ghany, M.;, Shamseldin, M. Fuzzy type two self-tuning technique of single neuron PID controller for brushless DC motor based on a COVID-19 optimization. *Int. J. Pow. Elec. Dr. Sys.* **2023**, 14(1), 562.
22. D. Baidya, S. Dhopte and M. Bhattacharjee, "Sensing System Assisted Novel PID Controller for Efficient Speed Control of DC Motors in Electric Vehicles," in *IEEE Sensors Letters* **2023**, *7*(1), 1-4.
23. Sorfina, U.; Islam, S.; Ching, K.; Soomro, D.; Yahaya, J. Adaptive position control of DC motor for brush-based photovoltaic cleaning system automation. *Bulletin Electr. Eng. Info.* **2023**, *12*(3), 1293–1301.
24. Mohanraj, N.; Kathirvelu, P.; Balasubramanian, R.; Sankaran, R.; Amirtharajan, R. Design of Permanent Magnet Brushless DC Motor Drive System for Energy Recouping in an Electric Automobile. *Arab J Sci Eng* **2023**, ().
25. Zhou, J.; Ebrahimi, S.; Jatskevich, J. Extended Operation of Brushless DC Motors Beyond 120° under Maximum Torque Per Ampere Control. *IEEE Transactions on Energy Conversion* (Early Access) **2023**, 1–12. doi: 10.1109/TEC.2023.3236594
26. Yang, W., Zhu, Y., Wang, C., Li, P., Liu, Y. (2023). Research on Simulation Model Control Strategy of Brushless DC Motor Based on MATLAB. In: Kountchev, R., Nakamatsu, K., Wang, W., Kountcheva, R. (eds) Proceedings of the World Conference on Intelligent and 3-D Technologies (WCI3DT 2022). Smart Innovation, Systems and Technologies, vol 323. Springer, Singapore.
27. Tripathi, R.; Singh, A.; Gangwar, P. Fractional order adaptive Kalman filter for sensorless speed control of DC motor, International Journal of Electronics **2023**, *110*(2), 373-390.
28. Saini, R.; Parmar, G.; Gupta, R. Chapter 4 - An enhanced hybrid stochastic fractal search FOPID for speed control of DC motor. *Fractional Order Systems and Applications in Engineering*. **2023**, 51-67.
29. Rahman, M.Z.U.; Leiva, V.; Martin-Barreiro, C.; Mahmood, I.; Usman, M.; Rizwan, M. Fractional Transformation-Based Intelligent H-Infinity Controller of a Direct Current Servo Motor. *Fractal Fract.* **2023**, *7*, 29.
30. Kumar, M.; Satheesh, G.; Peddakotla, S. Design of optimal PI controller for torque ripple minimization of SVPWM-DTC of BLDC motor. *Int. J. Pow. Elect. Dr. Sys.* **2023**, *14*(1), 283~293.
31. Mérida-Calvo, L.; Rodríguez, A.S.-M.; Ramos, F.; Feliu-Batlle, V. Advanced Motor Control for Improving the Trajectory Tracking Accuracy of a Low-Cost Mobile Robot. *Machines* **2023**, *11(1)*, 14.
32. Vered, Y.; Elliott; S. The use of digital twins to remotely update feedback controllers for the motion control of nonlinear dynamic systems. *Mechanical Systems and Signal Processing* **2023** 185, 109770.
33. Gurumoorthy, K.; Balaraman, S. Controlling the Speed of renewable-sourced DC drives with a series compensated DC to DC converter and sliding mode controller, *Automatika* **2023**, *64*(1), 114-126.
34. Moehle, N.; Stephen Boyd, S. Optimal current waveforms for brushless permanent magnet motors, *International Journal of Control* **2015**, *88*(7), 1389-1399
35. Niemeyer, Günter & Diolaiti, Nicola & Tanner, Neal.. Wave Haptics: Encoderless. *Virtual Stiffnesses* **2005**, 28. 22-33. 10.1007/978-3-540-48113-3_3.

36. Niemeyer, G., Diolaiti, N., Tanner, N. Wave Haptics: Encoderless Virtual Stiffnesses. In: Thrun, S., Brooks, R., Durrant-Whyte, H. (eds) *Robotics Research* **2007**. Springer Tracts in Advanced Robotics, vol 28. Springer, Berlin, Heidelberg.
37. Diolaiti N, Niemeyer G, Tanner NA. Wave Haptics: Building Stiff Controllers from the Natural Motor Dynamics. The International Journal of Robotics Research **2007**, *26*(1), 5-21.
38. Bernat, J.; Stepien, S. The adaptive speed controller for the BLDC motor using MRAC technique. *IFAC Proc.* Vol. **2011**, 44, 4143–4148.
39. Gowri, K.; Reddy, T.; Babu, C. Direct torque control of induction motor based on advanced discontinuous PWM algorithm for reduced current ripple. *Electr. Eng.* **2010**, 92, 245–255.
40. Rathaiah, M.; Reddy, R.; Anjaneyulu, K. Design of Optimum Adaptive Control for DC Motor. *Int. J. Electr. Eng.* **2014**, 7, 353–366.
41. Haghi, P.; Ariyur, K. Adaptive First Order Nonlinear Systems Using Extremum Seeking. In Proceedings of the 50th Annual Allerton Conference on Communication Control, Monticello, IL, USA, 1–5 October **2012**; pp. 1510–1516.
42. Available online: https://site.ieee.org/ias-idc/2019/01/29/prof-bob-lorenz-passed-away/ (accessed on 12 Dec 2022).
43. Zhang, L.; Fan, Y.; Cui, R.; Lorenz, R.; Cheng, M. Fault-Tolerant Direct Torque Control of Five-Phase FTFSCW-IPM Motor Based on Analogous Three-phase SVPWM for Electric Vehicle Applications. *IEEE Trans. Veh. Technol.* **2018**, 67, 910–919.
44. Apoorva, A.; Erato, D.; Lorenz, R. Enabling Driving Cycle Loss Reduction in Variable Flux PMSMs Via Closed-LoopMagnetization State Control. *IEEE Trans. Ind. Appl.* **2018**, 54, 3350–3359.
45. Flieh, H.; Lorenz, R.; Totoki, E.; Yamaguchi, S.; Nakamura, Y. Investigation of Different Servo Motor Designs for Servo Cycle Operations and Loss Minimizing Control Performance. *IEEE Trans. Ind. Appl.* **2018**, 54, 5791–5801.
46. Flieh, H.; Lorenz, R.; Totoki, E.; Yamaguchi, S.; Nakamura, Y. Dynamic Loss Minimizing Control of a Permanent Magnet Servomotor Operating Even at the Voltage Limit When Using Deadbeat-Direct Torque and Flux Control. *IEEE Trans. Ind. Appl.* **2019**, 3, 2710–2720.
47. Flieh, H.; Slininger, T.; Lorenz, R.; Totoki, E. Self-Sensing via Flux Injection with Rapid Servo Dynamics Including a Smooth Transition to Back-EMF Tracking Self-Sensing. *IEEE Trans. Ind. Appl.* **2020**, 56, 2673–2684.
48. Vidlak, M.; Gorel, L.; Makys, P.; Stano, M. Sensorless Speed Control of Brushed DC Motor Based at New Current Ripple Component Signal Processing. *Energies* **2021**, *14*, 5359.
49. Sands, T. Control of DC Motors to Guide Unmanned Underwater Vehicles. Appl. Sci. 2021, 11(5), 2144.
50. Shah, R.; Sands, T. Comparing Methods of DC Motor Control for UUVs. Appl. Sci. 2021, 11(11), 4972.
51. Koo, S.M.; Travis, H.; Sands, T. Impacts of Discretization and Numerical Propagation on the Ability to Follow Challenging Square Wave Commands. J. Mar. Sci. Eng. 2022, 10(3), 419.
52. Menezes, J.; Sands, T. Discerning Discretization for Unmanned Underwater Vehicles DC Motor Control. J. Mar. Sci. Eng. 2023, 11(2), 436.
53. Wang, Z.; Sands, T. Artificial Intelligence-Enhanced UUV Actuator Control. AI 2023, 4(1), 270-288.
54. Åström, K.; Apkarian, J.; Lacheray, H. Quanser Engineering Trainer (QET) Series: USB QICii Laboratory Workbook, DC Motor Control Trainer (DCMCT) Student Workbook. Available online: http://class.ece.iastate.edu/ee476/motion/Main_manual.pdf (accessed 13 February 2023).
55. Åström, K.;Wittenmark, B. Adaptive Control; Addison-Wesley: Boston, FL, USA, 1995.
56. Smeresky, B.; Rizzo, A.; Sands, T. Optimal Learning and Self-Awareness Versus PDI. Algorithms 2020, 13(1), 23.
57. Slotine, J.; Benedetto, M. Hamiltonian adaptive control on spacecraft. IEEE Trans. Autom. Control 1990, 35, 848–852.
58. Slotine, J.;Weiping, L. Applied Nonlinear Control; Prentice Hall: Englewood Cliffs, NJ, USA, 1991.
59. Sands, T. Virtual Sensoring of Motion Using Pontryagin's Treatment of Hamiltonian Systems. *Sensors* **2021**, 21, 4603.
60. Walker, A.; Putman, P.; Cohen, K. Solely Magnetic Genetic/Fuzzy-Attitude-Control Algorithm for a CubeSat. J. Space. Rock. 2015,52, 1627–1639.
61. Sands, T.; Bollino, K.; Kaminer, I.; Healey, A. Autonomous Minimum Safe Distance Maintenance from Submersed Obstacles in Ocean Currents. J. Mar. Sci. Eng. 2018, 6, 98.
62. 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.