DC Encoder Motor Control based on Koopman framework

Ravi Kiran Akumalla, Ravi Shankar Bahuguna and Tushar Jain

Abstract— **This work describes the hardware implementation of the Koopman framework on a DC motor with an encoder. Initially, the detailed procedure of finding the angular velocity of the DC motor by the Simulink® support package for Arduino is explained. Following this, the input voltage and output angular velocity data are collected by performing a step-response experiment on the DC motor. This data is used for the identification of approximate linear models of the DC motor applying the Koopman framework. Another data-driven model based on the MATLAB® System Identification toolbox is obtained using the same data for comparison. Both models are leveraged as predictors individually within the architecture of the linear model predictive controller (MPC) for angular velocity tracking of the DC motor. The Hardware-in-the-Loop (HIL) experiments are performed, and results are provided to show the feasibility of the Koopman framework.**

I. INTRODUCTION

DC motors are frequently utilised as actuating elements in many robotic and industrial applications due to their many adjustability options and ease of speed and position control [1]. In modelling a DC motor, the general approach is to neglect the non-linear effects and build a linear transfer function representation for the input-output relationship of the DC motor. This assumption is accurate enough and reliable when it comes to traditional control issues[2]. Neglecting the nonlinear effects such as magnetic saturation, varying load conditions, backlash, and friction in the motor can result in poor control performance and unacceptably high modelling errors [1][2]. Also, the parameters of the motor, such as armature resistance, inductance, etc., may change due to ageing effects in the motor. So, it is important to capture these parameter variations and nonlinearities for effective control of a DC motor. But, at a given time, we are left with only the access to input-output data from the motor. So if the mathematical model can be obtained instantaneously from these data, the control of the motor can be greatly improved.

System identification involves finding the mathematical model of the system by inserting specific input data and measuring the output data of the system. These techniques prove their effectiveness in controlling the DC motor by totally relying on the collected input-output data [3][4]. A scheme using system identification and PID auto-tuning is discussed for controlling the speed of a DC motor without knowing its specific parameters [5]. In fact, the data-driven identification and control of DC motors is vastly explored in the literature [6]. In contrast to previous works, we are exploring it using another data-driven technique known as the Koopman framework [7]. The main advantage of the Koopman framework is that the identified system is globally linear or linear over a wide range[8][9]. Literature shows a lot of work on the successful implementation of this framework [10][11][12]. But most of them are highly theoretical and complex to interpret and understand. So, in this work, we are demonstrating the implementation of this framework in a step-by-step manner by applying it to a simple mechatronic system (DC motor with encoder). Also, to prove its effectiveness and feasibility, the work is compared with another data-driven model obtained using the System Identification toolbox in MATLAB®.

The main contributions of this paper are given below:

- Application of the Koopman framework for data-driven identification and control of the DC motor and its hardware implementation using the Simulink® support package for Arduino.
- Comparison of the proposed method to the MATLAB® System Identification Toolbox-based data-driven identification and control of DC motor.

The rest of the paper is organised as follows. Section II describes the preliminaries of the Koopman framework, Koopman model predictive control (KMPC), circuit configuration for connecting the DC motor to Arduino, and angular velocity calculation from encoder pulses. In Section III, the methodology is described for data collection by step response, identification of linear Koopman predictor, identification of a data-driven transfer function model by the MATLAB®-based system identification toolbox for comparison, and hardware-in-loop implementation of the Koopman model predictive controller. Section IV explains the results, and Section V ends with discussing the conclusion and future works.

II. PRELIMINARIES

A. Notations

An identity matrix of order n is denoted by $I_{n\times n}$. A zero matrix with dimensions $m\times n$ is represented by $0_{m\times n}$. For a vector $a\in\mathbb{R}^n$, its transpose is denoted by a^T, time derivative by $\dot{a}$, and its pseudoinverse by $a^\dagger$. $||a||_Q^2$ represents the $Q-$ norm (i.e., a^TQa) and similarly for the R- norm, whereas $||a||_F$ represents the frobenius norm.

This work was not supported by any organization

R.K.Akumalla is with School of Mechanical and Materials Engineering, Indian Institute of Technology Mandi, HP, India d21040@students.iitmandi.ac.in

R.S.Bahuguna is with School of Computing and Electrical Engineering, Indian Institute of Technology Mandi, HP, India d23035@students.iitmandi.ac.in

T.Jain is with the Faculty of School of Computing and Electrical Engineering, Indian Institute of Technology Mandi, HP, India tushar@iitmandi.ac.in

B. Koopman Framework

The Koopman operator can be used to handle nonlinear systems by using a globally linear description [13]. A quick overview of the Koopman operator is described here; for a more detailed explanation, see [14]. Consider a nonlinear system with state $x \in R^n$ and input $u \in R^r$.

$$\dot{x} = f(x,u) \tag{1}$$

The state $x \in R^n$ in the nonlinear system (1) propagates over time by a nonlinear map $f : R^n \times R^r \rightarrow R^n$. Additionally, a collection of scalar-valued functions $g \in \mathscr{H}$ known as observable functions belong to the infinite-dimensional Hilbert space, $\mathscr{H}$ and are dependent on the states and inputs. Now, an infinite-dimensional linear Koopman operator $\mathscr{K} : \mathscr{H} \rightarrow \mathscr{H}$ is given as follows:

$$\mathscr{K} g(x,u) = g(f(x,u)) \tag{2}$$

In contrast to (1), this operator $\mathscr{K}$ allows the observable functions g to evolve linearly over time. But for practical applicability, a Koopman operator K of finite dimensions is approximately represented. An illustration of obtaining a finite-dimensional Koopman operator is shown in Fig. 1.

Given a nonlinear dynamical system described by (1), we collect $m+1$ samples of the input $u \in R^r$ and the state $x \in R^n$ at a fixed sampling interval. These, data samples are organised into the matrices:

- Input data: $U \in R^{r \times m}$
- State data: $X \in R^{n \times m}$ and $Y \in R^{n \times m}$.

It is to note that, Y is obtained by forward-shifting X i.e., $y(k) = x(k+1)$, where $y(k) \in Y$ and $x(k+1) \in X$. These gathered data matrices X and Y are lifted to higher dimensions using observable functions $g = [g_1, g_2 ..., g_p]^T$; $g_i : R^n \rightarrow R$ to capture the nonlinear behaviour in a linear fashion. These scalar-valued observable functions are chosen using machine learning approaches or by having a thorough understanding of the dynamics. The lifted state matrices are, $Z_X \in R^{p \times m}$ and $Z_Y \in R^{p \times m}$ where each column vector of Z_X is $z_{x(k)} = g(x(k))$; with $x(k) \in X$ being the kth sample. Lastly, an optimisation problem,

$$\begin{aligned} &\min_{A,B} ||Z_Y - AZ_X - BU||_F \\ &\min_{C} ||X - CZ_X||_F \end{aligned} \tag{3}$$

as shown in (3), is solved to obtain the linear Koopman operator $K = \begin{bmatrix} A & B \end{bmatrix}$. Here $A \in R^{p \times p}$ and $B \in R^{p \times r}$ define the linear evolution of the lifted state, and the lifted states are mapped back to the original states using $C \in R^{n \times p}$. Thus, the resultant Koopman linear predictor K approximates the nonlinear system via linear dynamics in the lifted space.

C. Koopman Model Predictive Control

The obtained Koopman predictor is used as the prediction model within the MPC architecture [15]. This implementation of the MPC controller with a Koopman predictor is referred to as Koopman-based linear MPC (KMPC) for nonlinear systems.

$$\begin{aligned} \min_{u(k:k+N-1)} \quad & \sum_{j=k}^{k+N-1} ||x(j) - x_{ref}||_{\boldsymbol{Q}}^2 + ||u(j)||_{\boldsymbol{R}}^2 \\ \text{s.t.} \quad & z(j+1) = Az(j) + Bu(j) \\ & x(j) = Cz(j) \\ & u_{min} \leq u(j) \leq u_{max} \\ & x_{min} \leq x(j) \leq x_{max} \end{aligned} \tag{4}$$

In this KMPC formulation (4), $x(j)$ and x_{ref} represents the current state and desired state, respectively. The input u is calculated by minimising the cost function satisfying the constraints. The minimum and maximum constraints on the input are u_{min}, u_{max} and on the state are x_{min}, x_{max} respectively, whereas A, B, C are Koopman linear predictor matrices.

D. Circuit configuration of DC Motor with encoder

The circuit diagram for connecting the encoder motor with the Arduino Mega 2560 is given in Fig. 2. The DC motor has six connections. The motor connections (M_1, M_2) are connected to the L298 motor driver, which is powered by a 12V external supply. The encoder power supply connections V_{cc}, GND are connected to 5V and GND pins of Arduino, and the encoder pulse pins c_1, c_2 are connected to external interrupt pins of Arduino. Both the readings from c_1, c_2 are required for calculating the position of the shaft, while for calculating angular velocity, either of the pulses from c_1, c_2 can be used. So, c_1 is connected to one of the external interrupt pins (pin 21) in the Arduino Mega 2560 and c_2 is left with no connection. The Simulink® support package for Arduino is installed in the computer to which Arduino Mega 2560 is connected and configured to use. The pin 5 of the Arduino is connected to the motor driver to vary the input voltage to the DC motor by varying the duty cycle of pulse width modulation (PWM) signal.

E. Angular Velocity calculation by encoder pulses

The calculation of angular velocity of a DC motor using the Arduino + Simulink® support package is explained in this section. The mathematics behind converting the encoder pulses to angular velocity is as below:

$$counts(t) = \text{Rising edge pulses from Encoder}$$

$$counts/second = \frac{counts(t) - counts(t-k)}{k}$$

$$RPS = \frac{counts/second}{counts/rotation}$$

$$RPM = RPS * 60$$

From Fig. 3, we can see an external interrupt block continuously stimulating a function-call subsystem block based on pulses from the encoder sensor. These stimuli by the interrupt block can be set to rising edge, falling edge, or change in edge of received encoder pulses. In this work, the rising edge is considered, so the function-call subsystem keeps on adding one count to the already existing counts whenever there is a

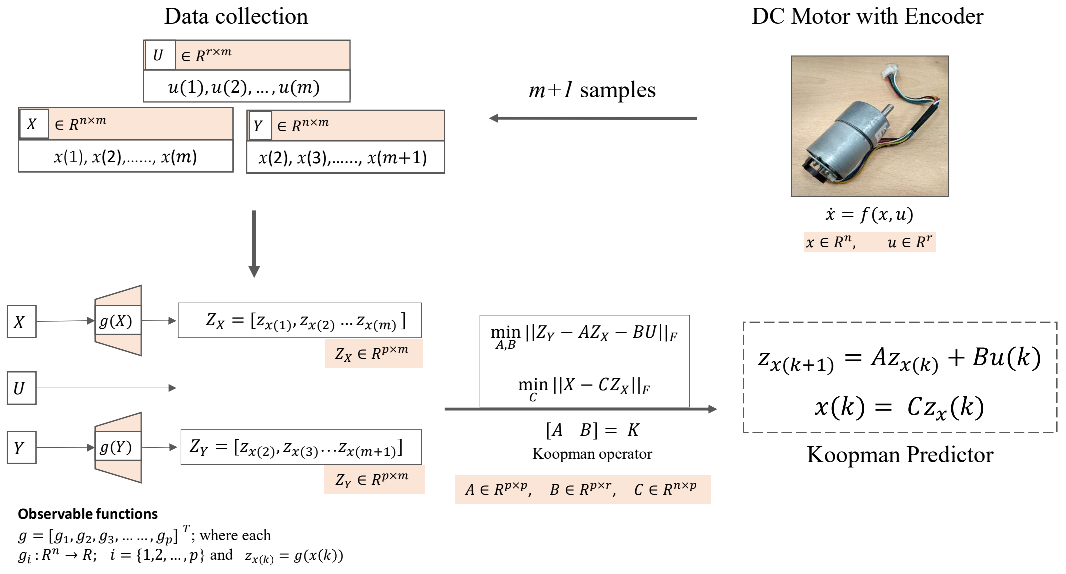
Data collection
DC Motor with Encoder
$U \in R^{r\times m}$
$u(1), u(2), \dots, u(m)$
$X \in R^{n\times m}$
$x(1), x(2), \dots, x(m)$
$Y \in R^{n\times m}$
$x(2), x(3), \dots, x(m+1)$
$m+1$ samples
$\dot{x} = f(x,u)$
$x \in R^n, \quad u \in R^r$
$g(X)$
$Z_X = [z_{x(1)}, z_{x(2)} \dots z_{x(m)}]$
$Z_X \in R^{p\times m}$
$g(Y)$
$Z_Y = [z_{x(2)}, z_{x(3)} \dots z_{x(m+1)}]$
$Z_Y \in R^{p\times m}$
$\min_{A,B} \|Z_Y - AZ_X - BU\|_F$
$\min_{C} \|X - CZ_X\|_F$
$[A \quad B] = K$
Koopman operator
$A \in R^{p\times p}, \quad B \in R^{p\times r}, \quad C \in R^{n\times p}$
$z_{x(k+1)} = Az_{x(k)} + Bu(k)$
$x(k) = Cz_x(k)$
Koopman Predictor
Observable functions
$g = [g_1, g_2, g_3, \dots, g_p]^T$; where each
$g_i : R^n \to R; \quad i = \{1,2,\dots,p\}$ and $z_{x(k)} = g(x(k))$

Fig. 1. Koopman framework

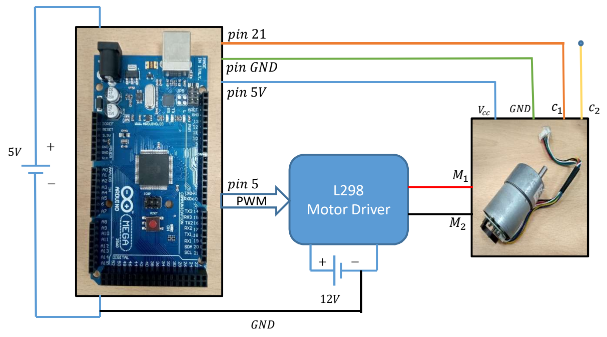

Fig. 2. Circuit diagram

high rising edge on c_1. A rate transition block is inserted to handle the different operating data rates between the blocks. These counts $(counts(t))$ at a given time are subtracted from their previous value by delaying $counts(t)$ an instant of 'k' to obtain $counts(t-k)$. This difference is divided by the delayed instant value 'k' to obtain counts per second $(counts/sec)$. The obtained $counts/sec$ are divided by counts per rotation $(counts/rotation)$ to obtain the RPS and then multiplied with 60 to obtain the RPM finally. RPM and RPS are rotations per minute and rotations per second, respectively. A low-pass filter is introduced to decrease the high-frequency noise in the output RPM value.

The DC motor is rated at 12 V, 260 RPM. The gear ratio is 160:1. From experiments, it is observed that the encoder's rising pulses for one full rotation of the outer shaft are 480, whereas the rising pulses for one full rotation of the encoder disc are 3. The whole blocks in Fig. 3 are made into a subsystem block, named as Encoder.

III. Methodology

A. Data collection by Step response of DC motor

For data collection in the Koopman framework, the input voltage (u) and output angular velocity (x) are collected by performing the step response of the DC motor for 10 seconds. The DC voltage of $12V$ is given as input, and angular velocity (RPM) is collected as output at a sampling time of 0.1 seconds. Fig. 4 shows the step response model in Simulink®. A PWM signal of full duty cycle is generated from Pin 5 of the Arduino using the PWM block in the support package. It receives values from 0-255, being an 8-bit converter, by which the duty cycle can be varied from 0-100%. This Pin 5 is connected to the L298 motor driver to provide maximum input to the DC motor. Using a digital multimeter, it is found that the maximum input voltage to the DC motor at full duty cycle is 11.84V. So a gain block $(11.84/255)$ is used to record the approximate input voltage based on PWM input to the motor driver. The collected data is arranged into state data matrices $X \in R^{1\times 100}$, $Y \in R^{1\times 100}$ and input data matrix $U \in R^{1\times 100}$.

B. Identification of Linear Koopman Predictor

The collected data is passed through observable functions, and the Koopman framework is implemented as discussed in Section II-B. It is to be noted that the better approximation of the Koopman linear predictor depends on the selection of these observable functions. These are generally selected as standard basis functions (monomial, polynomial, trigonometric, etc.) [15], selected from the dynamics [16], or through deep learning techniques [11]. The observable functions selected in this work are shown in the Table I. State data matrices $X \in R^{1\times 100}$, $Y \in R^{1\times 100}$ are thus lifted to Z_X, and Z_Y. From these lifted matrices, the Koopman linear matrices A, B, and C are identified by solving the

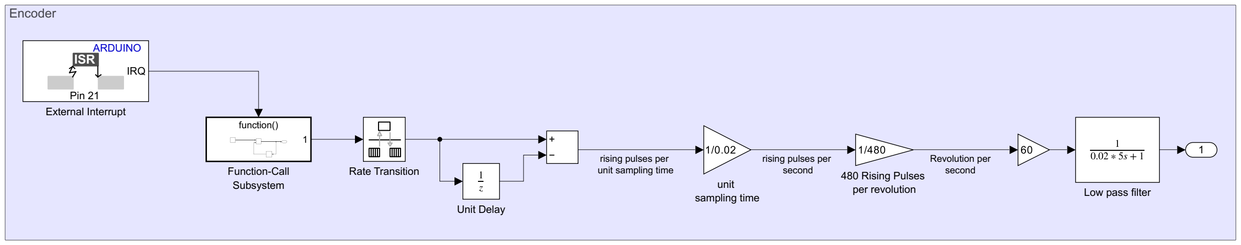

Fig. 3. RPM calculation using Encoder

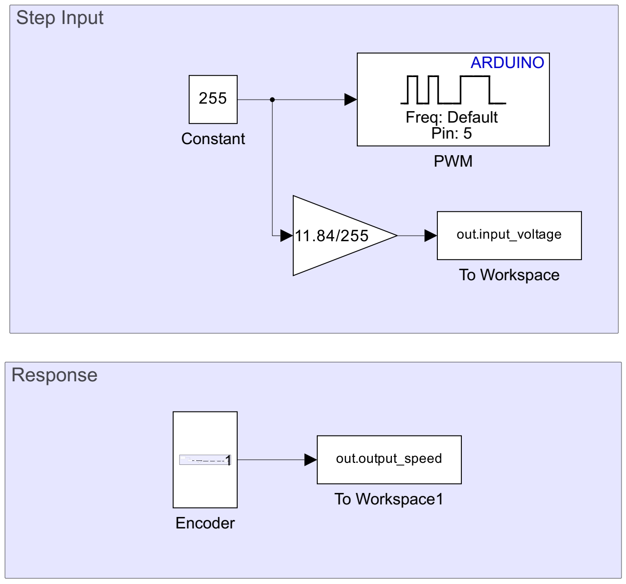

Fig. 4. Step response

TABLE I
OBSERVABLE FUNCTIONS

	g	$Z_X, Z_Y \in$
Koopman$_1$	x	$R^{1\times 100}$
Koopman$_2$	$[x, \sin x, \cos x]^T$	$R^{3\times 100}$
Koopman$_3$	$[x, x^2, x^3]^T$	$R^{3\times 100}$

optimisation problem (3) analytically as:

$$K = \begin{bmatrix} A & B \end{bmatrix} = MG^{\dagger}$$
$$\text{where } M = \frac{1}{m+1}\Sigma_{k=1}^{m}\left[z_y(k)\right]\begin{bmatrix} z_x(k) \\ u(k) \end{bmatrix}^T \tag{5}$$
$$G = \frac{1}{m+1}\Sigma_{k=1}^{m}\begin{bmatrix} z_x(k) \\ u(k) \end{bmatrix}\begin{bmatrix} z_x(k) \\ u(k) \end{bmatrix}^T$$

$z_y(k)$, $z_x(k)$, $u(k)$ are the k^{th} column vectors of Z_Y, Z_X and U matrices, respectively. As the angular velocity state (x) is augmented into the lifted state, it can be obtained using the following C matrix.

$$C = 1;\ \text{for Koopman}_1$$
$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix};\ \text{for Koopman}_{2,3} \tag{6}$$

C. Using the System Identification toolbox in MATLAB®

The identified data-driven Koopman linear models from the previous section are compared with another data-driven transfer function estimated using *the System Identification toolbox* in MATLAB®. The toolbox relies on a nonlinear least squares algorithm to estimate the transfer function [17]. To identify the model using this toolbox, the MATLAB® commands are given as below.

```
data = iddata(y,u,Ts);
  TF = tfest(data,p,z);
```

y is the output data matrix, u is the input data matrix, and Ts is sampling time. The command $tfest$ estimates the transfer function with p poles and z zeros. The general transfer function of the DC motor is of second order with 2 poles. So, considering 2 poles ($p = 2, z = 0$), the transfer function (TF) is obtained at a sampling time of 0.1 sec ($Ts = 0.1$), using the same angular velocity data and voltage data as output (y) and input data (u) matrix, respectively.

The identified Koopman$_{1,2,3}$ models and TF model are validated by giving the same step input and compared with the true response of the motor. Once the results are found satisfactory, they are used in the architecture of the model predictive controller to calculate the optimal control inputs to the DC motor.

D. Implementation of Model Predictive Controller

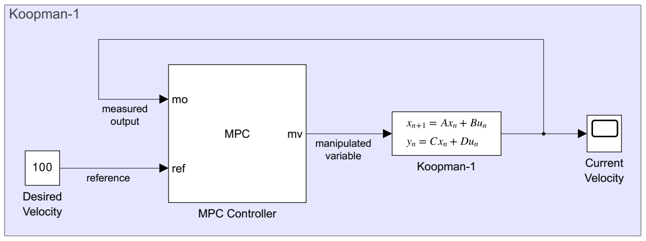

Fig. 5. Model predictive control with data-driven predictor

Upon finding the data-driven linear models, they are leveraged in the MPC architecture (individually) as predictors. The MPC controller block in Simulink®, provides a smooth interface for implementation of MPC [18], [19]. Initially, the 'mpc' objects for the predictors are designed using correspondingly obtained linear models, as shown in Fig. 5. The MPC controller block internally designs the 'mpc' objects which contains the information of the prediction model, the quadratic cost function as discussed in (4) and the constraints specified by the user. These 'mpc' objects are used in the corresponding MPC controller blocks during the hardware in-loop implementation. Fig. 6 shows the Simulink® model for HIL implementation of MPC. The constraints on the input voltage are set as $u \in [0, 12]$ and on the output angular velocity are set as $x \in [0, 260]$.

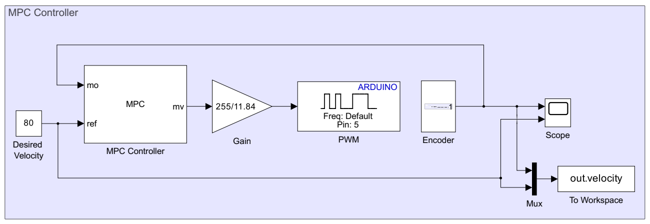

Fig. 6. Simulink® model for HIL implementation of MPC

IV. Results and Discussion

To start with data collection, the step response experiment is performed on the motor. Fig. 7 shows the input-output data collected from the step response. The Koopman linear models and system identification-based model (TF) are obtained, as discussed above in Section III, from these input-output data. Then an open-loop comparison is made among

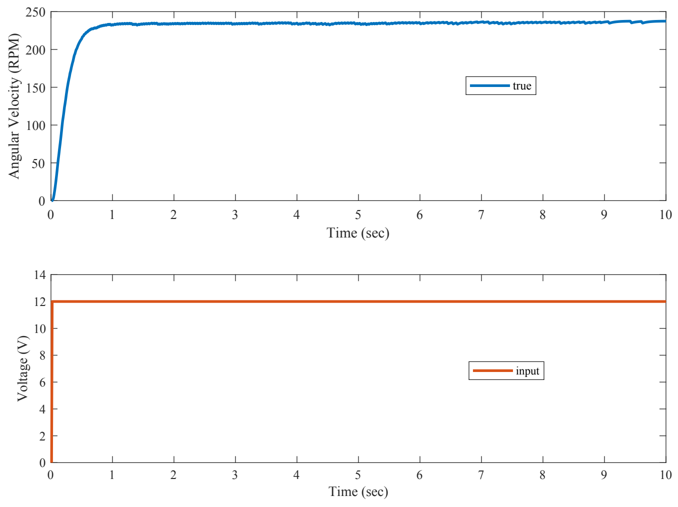

Fig. 7. Input-Output data collected from Step response

the models with respect to the true velocity of the motor. The same input voltage is applied to all the models along with the DC motor. Fig. 8 shows the response of the identified models in comparison with the true velocity. The open-loop prediction error can be observed in Fig. 9. It can be inferred that the transient response characteristics are better captured by the TF model, while the steady-state response is better captured by Koopman-based models. Further, the root mean square error (RMSE) for the open-loop response of all the models is found to be 9.085, 10.44, 9.93, and 9.42 RPM for Koopman$_{1,2,3}$ and TF models, respectively. As the resultant models are satisfactory, they are used as predictors in the model predictive controller as discussed in section III-D.

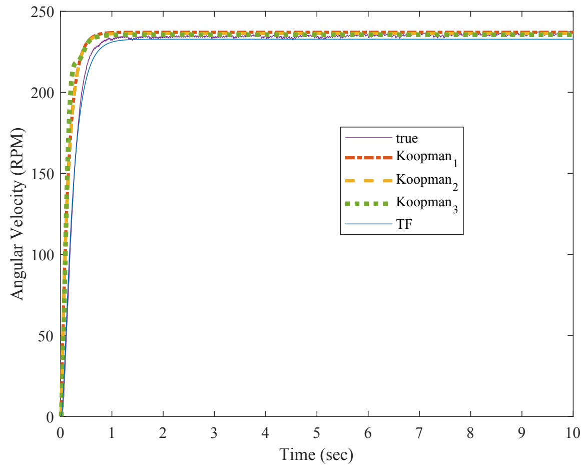

Fig. 8. Comparison of open-loop predicted responses with true response

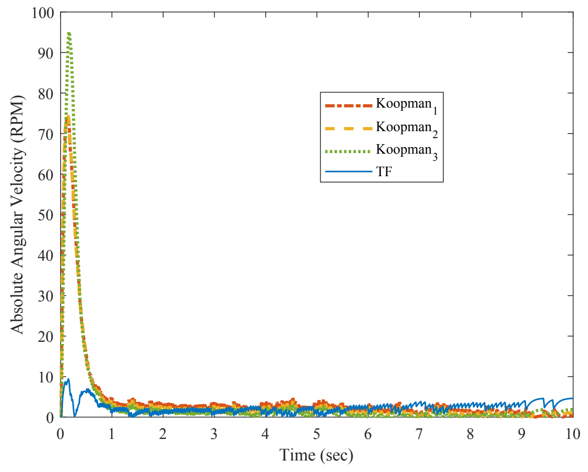

Fig. 9. Open loop prediction error

Fig. 10 shows the MPC-based angular velocity tracking of 50 RPM for all the data-driven predictors. The corresponding closed-loop input voltage can be observed in Fig. 11.

The quantitative analysis of closed-loop response, using the identified data-driven models as predictors in MPC, is performed considering the rise time (t_r), settling time (t_s), percentage overshoot (ϕ), and RMSE as shown in Table II.

TABLE II
Analysis of Closed Loop Response

	t_r (sec)	t_s (sec)	$\phi(\%)$	RMSE (RPM)
Koopman$_1$	0.1303	1.2062	27.29	3.88
Koopman$_2$	0.1309	1.2323	30.26	4.52
Koopman$_3$	0.1464	0.8068	15.44	4.20
TF	0.1366	0.8238	15.11	4.14

It can be observed that the Koopman$_1$, Koopman$_2$ models are having a higher overshoot with a faster rise time.

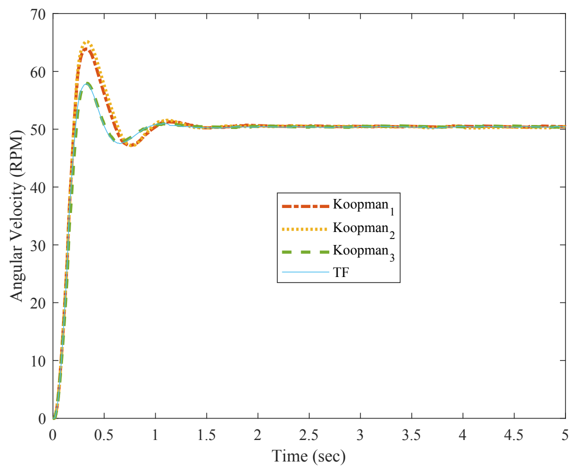

Fig. 10. MPC-based tracking of Angular velocity

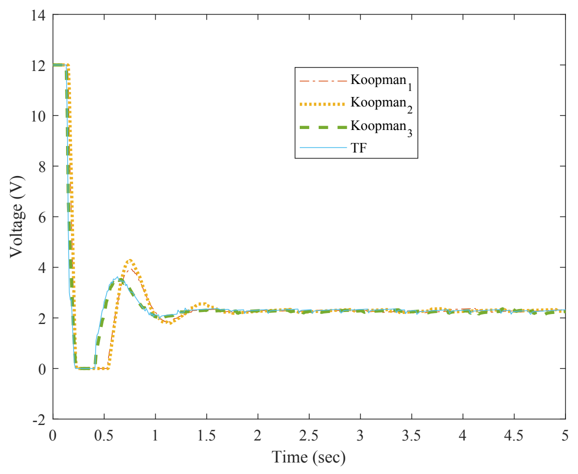

Fig. 11. Closed loop input voltage

While the Koopman$_3$ and TF models have lower overshoot and approximately similar closed-loop characteristics. The RMSE of the Koopman models shows that the steady state performance got better, although with increased overshoots in the transient stage.

V. Conclusion and Future works

The paper thus demonstrates the hardware implementation of the Koopman framework using MATLAB® and Simulink® Support Package for Arduino. The angular velocity tracking of the DC motor using the data-driven models obtained by the Koopman framework is performed step by step with HIL experiments. The obtained model is then compared with a well-established data-driven transfer function model obtained by the system identification toolbox in MATLAB® to support our results. It can be observed that the performance of the Koopman framework-based control of the DC motor varies with the selection of observable functions. Further, the steady-state results using Koopman models are better than those of the transfer function model. These results can be further improved by selecting a better set of observable functions. So, in future works, the deep learning-based Koopman framework will be adopted for deriving the observable functions instead of selecting the standard observable functions.

References

[1] J.-H. Horng, "Neural adaptive tracking control of a dc motor," *Information sciences*, vol. 118, no. 1-4, pp. 1–13, 1999.

[2] S. E. Lyshevski, "Nonlinear control of mechatronic systems with permanent-magnet dc motors," *Mechatronics*, vol. 9, no. 5, pp. 539–552, 1999.

[3] T. Kara and I. Eker, "Nonlinear modeling and identification of a dc motor for bidirectional operation with real time experiments," *Energy Conversion and Management*, vol. 45, no. 7-8, pp. 1087–1106, 2004.

[4] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, pp. 459–471, 2007.

[5] W.-J. Tang, Z.-T. Liu, and Q. Wang, "Dc motor speed control based on system identification and pid auto tuning," in *2017 36th Chinese Control Conference (CCC)*. IEEE, 2017, pp. 6420–6423.

[6] B. Arifin, A. A. Nugroho, B. Suprapto, S. A. D. Prasetyowati, and Z. Nawawi, "Review of method for system identification on motors," in *2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*. IEEE, 2021, pp. 257–262.

[7] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[8] A. Mauroy, Y. Susuki, and I. Mezić, *Koopman operator in systems and control*. Springer, 2020.

[9] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing koopman theory to allow for inputs and control," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018.

[10] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[11] Y. Xiao, X. Zhang, X. Xu, X. Liu, and J. Liu, "Deep neural networks with koopman operators for modeling and control of autonomous vehicles," *IEEE transactions on intelligent vehicles*, vol. 8, no. 1, pp. 135–146, 2022.

[12] B. Sharan, A. Dittmer, and H. Werner, "Real-time model predictive control for wind farms: a koopman dynamic mode decomposition approach," in *2022 European Control Conference (ECC)*. IEEE, 2022, pp. 1006–1011.

[13] P. Bevanda, S. Sosnowski, and S. Hirche, "Koopman operator dynamical models: Learning, analysis and control," *Annual Reviews in Control*, vol. 52, pp. 197–212, 2021.

[14] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PloS one*, vol. 11, no. 2, p. e0150171, 2016.

[15] M. Korda, Y. Susuki, and I. Mezić, "Power grid transient stabilization using koopman model predictive control," *IFAC-PapersOnLine*, vol. 51, no. 28, pp. 297–302, 2018.

[16] R. K. Akumalla, A. Shukla, and T. Jain, "Koopman framework-based observer design for actuator fault identification in quadrotor," in *2024 Tenth Indian Control Conference (ICC)*. IEEE, 2024, pp. 344–349.

[17] R. K. Akumalla and T. Jain, "Online tuning of koopman operator for fault-tolerant control: A case study of mobile robot localising on minimal sensor information," *Machines*, vol. 13, no. 6, 2025. [Online]. Available: https://doi.org/10.3390/machines13060454

[18] MathWorks, *mpcController*, The MathWorks, Inc., 2025, accessed: 2025-05-26. [Online]. Available: https://in.mathworks.com/help/mpc/ref/mpccontroller.html

[19] ——, *QP Solver Used in Model Predictive Control Toolbox*, The MathWorks, Inc., 2025, accessed: 2025-05-26. [Online]. Available: https://in.mathworks.com/help/mpc/ug/qp-solver.html