Real-Time Data-Driven System Identification of Motor Drive Systems Using Online DMDc

Muhammed Ali Gultekin, *Student Member, IEEE*
Electrical and Computer Engineering Department
University of Connecticut
Storrs, CT
muhammed_ali.gultekin@uconn.edu

Ali Bazzi, *Senior Member, IEEE*
Electrical and Computer Engineering Department
University of Connecticut
Storrs, CT

2022 IEEE Energy Conversion Congress and Exposition (ECCE) | 978-1-7281-9387-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ECCE50734.2022.9947577

Abstract—**Real-time parameter and state estimation is an interesting topic in motor drive applications, especially with more autonomous systems employing motor drives. With no parameter dependency, data-driven methods have become more popular. One of these methods is dynamic mode decomposition (DMD). Its controllable variant, DMDc, and online variant, online-DMDc, are of interest in this paper. DMD and DMDc require large datasets for training, but ODMDc requires smaller datasets to be initialized. The comparisons for these algorithms are made by looking into MATLAB/ Simulink simulations and experiments. It is shown that DMD and DMDc are powerful algorithms but online implementation is comparable to the offline versions in terms of estimation performance.**

Keywords—data-driven estimation, system identification, induction motor drives

I. Introduction

Dynamical systems are prone to disturbances, noise, and faults that can alter system behavior and thus alter the mathematical model of the plant. This also applies to power electronics and motor drives. Any closed-loop system with a model-based controller or estimator relies on healthy operating conditions unless a fault-tolerant control technique is applied to mitigate faults or model variations. This can be achieved through real-time system identification and estimation.

Building an estimator for real-time system identification applications can be challenging. In addition to uncertainty contributed by system model and noisy measurements, estimation adds memory and computational complexity. These estimators can be model-based, where the underlying dynamics of the model are known. Such estimators can be deterministic such as state observers, or they can be probabilistic such as Kalman filters.

The topic of estimation in machines and drives is not novel, there are established methods to estimate system parameters and states [1-4]. However, most of these methods are dependent on machine parameters. Having a data-driven and parameter-free estimation tool is thus of interest as it eliminates model inaccuracies when the system sees unexpected or uncertain disturbances.

There are many system identification methods reported in the literature but in this paper, online implementation of dynamic mode decomposition (DMD) and its extension to dynamic mode decomposition with control (DMDc) is considered. Online-DMDc (ODMDc) is compared to DMDc in [5] with system-theoretic considerations but with no implementation in power electronic systems. Previously, DMDc is introduced to the power electronics community in [6], and it is taken as the foundation for the work presented here where real-time implementation considerations, such as the computational burden and parameter selection, are investigated. Simulation results from a MATLAB/Simulink model are shown for different cases and are implemented on a real-time platform as a proof of concept.

This work is supported by the National Science Foundation under Award 1752297.

In the following section of the paper, the basics of DMD, DMDc, and ODMDc are explained and applied to an induction motor drive. In Section III, simulations for different cases are shown. Section IV shows experimental results for validation. Section V compares the algorithms and results, and discusses implementation considerations. Section VI concludes the paper with potential applications and future work.

II. Background

Before delving into the algorithm derivation, it is useful to investigate the target system. A voltage-source inverter (VSI)-fed induction motor (IM) with indirect field-oriented control (IFOC) is considered in this study where the high-level block diagram is given in Figure 1(a). The IM model uses flux-based differential equations which are summarized in the equation set (1) [9]. In (1); $\psi, v, L,$ and R represent flux, voltage, inductance, and resistance respectively. Subscripts q and d are qd components of these quantities whereas s represents a stator variable and r represents a rotor variable. ω_s and ω_r are slip and rotor frequencies. L_m is the mutual inductance, P is the number of poles, J is the inertia and T_L is the load torque.

$$\begin{aligned}
\dot{\psi}_{qs} &= -\frac{R_s}{L_s - L_m^2/L_r}\psi_{qs} - \omega_s\psi_{ds} - \frac{R_s}{L_m - L_sL_r/L_m}\psi_{qr} + v_{qs}\\
\dot{\psi}_{ds} &= \omega_s\psi_{qs} - \frac{R_s}{L_s - L_m^2/L_r}\psi_{ds} - \frac{R_s}{L_m - L_sL_r/L_m}\psi_{dr} + v_{ds}\\
\dot{\psi}_{qr} &= -\frac{R_r}{L_m - L_sL_r/L_m}\psi_{qs} - \frac{R_r}{L_r - L_m^2/L_s}\psi_{qr} - (\omega_s - \omega_r)\psi_{dr}\\
\dot{\psi}_{dr} &= -\frac{R_r}{L_m - L_sL_r/L_m}\psi_{ds} + (\omega_s - \omega_r)\psi_{qr} - \frac{R_r}{L_r - L_m^2/L_s}\psi_{dr}\\
J\dot{\omega}_r &= \frac{3P}{2(L_m - L_sL_r)/L_m}(\psi_{ds}\psi_{qr} - \psi_{qs}\psi_{dr}) - T_L
\end{aligned} \tag{1}$$

For experiments, a 1.5hp IM is supplied with a VSI, and its rotating shaft is coupled with a dynamometer for load variation. The digital control and estimation platform is dSpace and is shown in Fig. 2.

Consider a discrete-time dynamical system with a system matrix A, an input matrix B, a state vector $x[n]$, and an input vector $u[n]$, where n is the sample time. The DMD algorithm considers $u[n]$=0 and DMDc takes the control input $u[n]$ into consideration. A mapping can be found from $x[n]$ to $x[n+1]$ as shown in (2). The next step model can be given as a function of the current state and the input as shown in (3). By utilizing matrix manipulation and time-shifting (assuming matrix G is time-invariant), we can follow the steps described in equations (4)-(6). In most cases, the matrix Υ is not a square matrix, hence its inverse cannot be taken directly. In this case, the pseudo inverse is used and is denoted by the superscript (†). The final equation can be put as in equation (7) where X_1 is $X[2{:}\ n]$ and X_2 is $X[1{:}n\text{-}1]$.

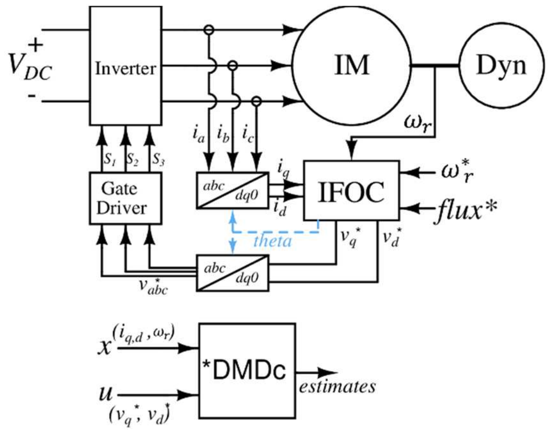


Figure 1: Block diagram of the system. The induction motor is driven by a voltage source inverter and controlled with IFOC.

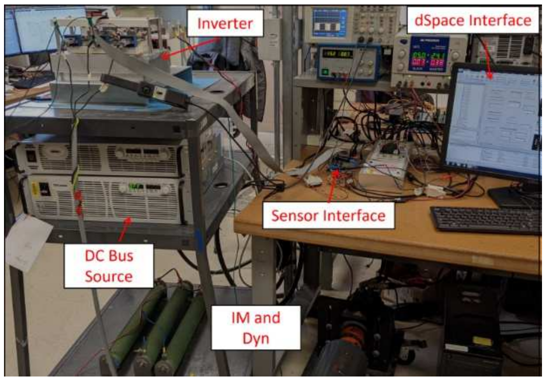


Figure 2: Hardware setup consisting of induction motor and dyne set, inverter driven by power supply and dSpace controller

$$x[n+1] = A\,x[n] + B\,u[n] \tag{2}$$

$$x[n+1] = f(x[n], u[n]) \tag{3}$$

$$G = [A \;\; B], \Upsilon[n] = \begin{bmatrix} x[n] \\ u[n] \end{bmatrix} \tag{4}$$

$$x[n+1] = \; G\Upsilon[n] \tag{5}$$

$$G \; = \; x[n+1]\Upsilon^{-1}[n] \tag{6}$$

$$G \; = \; x[n]\Upsilon^{\dagger}\,[n-1] \tag{7}$$

$$\Upsilon = \begin{bmatrix} X_2 \\ U_2 \end{bmatrix}, \qquad G = X_1\Upsilon^{\dagger} \tag{8}$$

These derivations can be seen in more detail in [6]. The original DMD [7] and DMDc [8] methods are system identification methods, and they are not suited for online and adaptive applications. As we get new data, the matrices X_1, X_2 and U_2 would grow unboundedly. If these algorithms are to be implemented online, then a 'window' is required to have a finite amount of data. The authors in [6] gave detailed analyses on windowed DMDc (WDMDc). But WDMDc requires significant computational and memory resources, which burdens the embedded system.

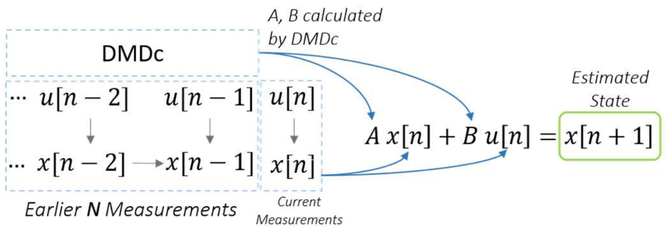
Online-DMD is a variation of DMD for streaming data and real-time updates to matrix G [5]. It aims to reduce the computational effort of the DMD using recursive algorithms. In this paper, the idea of the Online-DMD is applied to DMDc (non-windowed) and the authors propose an update algorithm as follows.

Start with collecting enough data to calculate initial G and P using (9), and update G and P using (10). Derivation of (10) is not trivial, it is produced using recursive least squares regression and matrix manipulations.

$$G = X_1\Upsilon^{\dagger}, \qquad P = (\Upsilon\,\Upsilon^{\mathrm{T}})^{-1} \tag{9}$$

$$G_{new} = G_{old} + (V - G_{old}U)\Gamma_{k+1}U^{T}P_k \tag{10}$$

where V, U, and Γ are intermediate matrices [5]. The main advantage of this approach is the significant reduction of computation and memory requirements. This algorithm will be referred as online DMDc (ODMDc). After identifying the system using past data, the next states can be estimated using the identified model, which can be considered as predictive modeling. Predictive modeling is explained visually in Figure 3. The calculation of A and B matrices can be done using DMD, DMDc or ODMDc algorithms; but, to estimate the next state, the current measurements and control inputs are being used with the calculated A and B matrices.



A, B calculated by DMDc
DMDc
Estimated State
$\cdots\ u[n-2] \quad u[n-1] \quad u[n]$
$A\,x[n] + B\,u[n] = x[n+1]$
$\cdots\ x[n-2] \rightarrow x[n-1] \quad x[n]$
Earlier N Measurements
Current Measurements

Figure 3. Predictive modeling process for DMDc

III. Simulations for Different Operating Conditions

To make the state estimations and identify the system; DMD, DMDc and ODMDc methods are implemented in MATLAB/Simulink for a VSI-fed induction motor drive setup. The proposed system is given in Figure 1. Two test cases are constructed to reflect different speed and load conditions. In the first case, step-type speed reference changes are considered with a quadratic load condition. The load formulation is given in (11) where T_{load} is the load torque in N.m and ω_r is the rotor speed in rad/s. The speed reference is set to 1200 rpm at the start and it is changed to various values ranging between 800 rpm to 1800 rpm throughout the 25 s simulation. The simulation results for this case are given in Figure 4.

$$T_{load} = 0.9 * 10^{-4} * \omega_r^2 \tag{11}$$

The second case has ramp type changes which are introduced to both increment and decrement reference changes. Speed reference starts from 1200 rpm and then it takes values between 500 rpm and 1500 rpm. The load also dynamically varies with sharp step changes between 0-5N.m. The results for this case are shown in Figure 5.

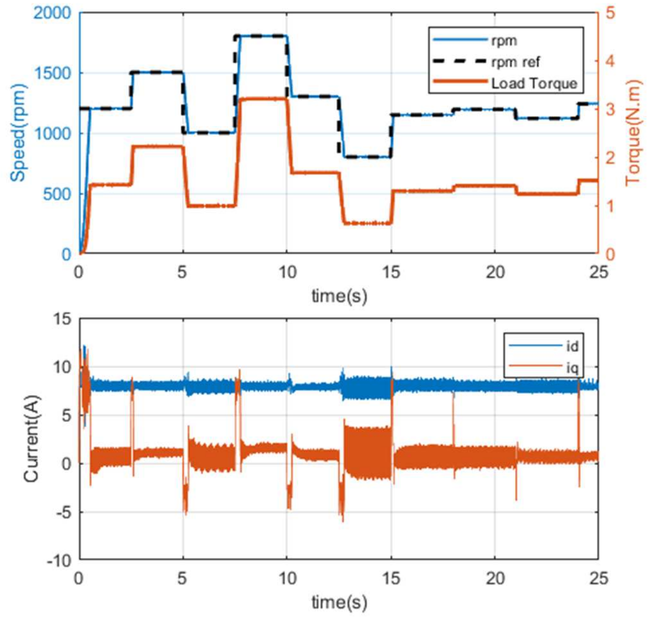


Figure 4: Simulation results for the first test case. Top: speed of the motor, speed reference and load torque. Bottom: Stator currents, d and q components.

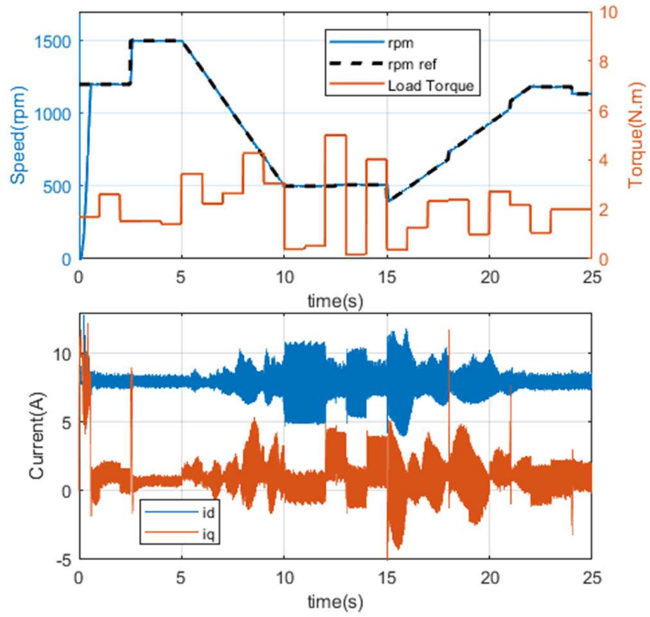


Figure 5: Simulation results for the second test case. Top: speed of the motor, speed reference and load torque. Bottom: Stator currents, d and q components.

The reason to have two different scenarios is to test models generated by the DMD, DMDc and ODMDc algorithms. Loading and the reference types are drastically different in the two scenarios but they are applied to the same system. A good model should estimate states regardless of the operating conditions.

The results from the first case are used as training or initialization data. The term *training data* is used for offline estimators DMD and DMDc whereas the term *initialization data* is for online estimator ODMDc. For both estimators, models are generated and these models are tested using the results from the second simulation case. The online estimator ODMDc is initialized with a small portion of the data of the first simulation case. And then the model is updated in each sampling time, resulting in an adaptive model. Figure 6 summarizes this procedure.

Estimation results for DMD, DMDc and ODMDc are given in Figures 7-9. Figure 7 shows the estimation of i_d and estimation errors or residuals for each estimation. Figure 8 shows the estimation for i_q and resulting residuals, and Figure 9 shows estimations for speed as well as resulting residuals. The plotted residuals are not normalized and are calculated using (12) where x represents a state and $\hat{x}$ represents an estimated state.

$$residual = x - \hat{x} \tag{12}$$

Figures 7-9 show clearly that a well-trained model can accurately estimate under different operating and load conditions. DMD and DMDc have almost identical results whereas ODMDc has slightly different results. It should be noted that the data quality of the simulations is perfect, there is no noise or any data loss while obtaining data.

The A and B matrices are generated by the estimators and compared. Since they generate similar results, they should be close to one another. To compare the models, eigenvalues of the A matrices can be compared.

$$A_{dmd} = \begin{bmatrix} 0.975 & 0.013 & 1.45e^{-4} \\ 0.0056 & 0.951 & -1.26e^{-6} \\ -0.0243 & 0.373 & 0.999 \end{bmatrix} \tag{13}$$

$$\lambda_{1,2,3-DMD} = [1, 0.978 ,0.948] \tag{14}$$

$$A_{dmdc} = \begin{bmatrix} 0.9702 & 0.0081 & 2.47e^{-4} \\ -0.0036 & 0.9220 & -0.0044 \\ -0.0466 & 0.3331 & 0.9961 \end{bmatrix} \tag{15}$$

$$\lambda_{1,2,3-DMDc} = [0.974, 0.957 \ \pm 0.014i] \tag{16}$$

As can be seen from (13) and (14), even though the estimations are close to one another, the eigenvalues are different. The model generated by the ODMDc is not static like DMD or DMDc as changes in every iteration. The table below shows the evolution of the eigenvalues of ODMDc from initialization to the end. Table I shows that as ODMDc evolves, two eigenvalues also change from a complex conjugate to real values. The nature of the incoming data shapes the eigenvalues as well. One important point is the stability of the algorithm, where the eigenvalues did not move outside of the unit circle.

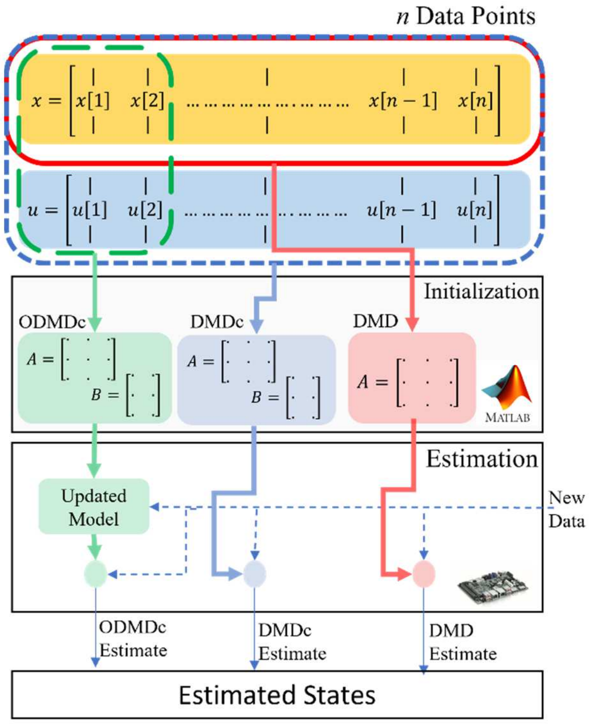


Figure 6: Estimation procedure. Large amounts of data are necessary to train DMD and DMDc, a smaller portion is enough for the ODMDc. The initialization or training portion is done offline in MATLAB. After models are generated, the estimation stage can be performed in an embedded platform. The estimation procedure is performed as (2).

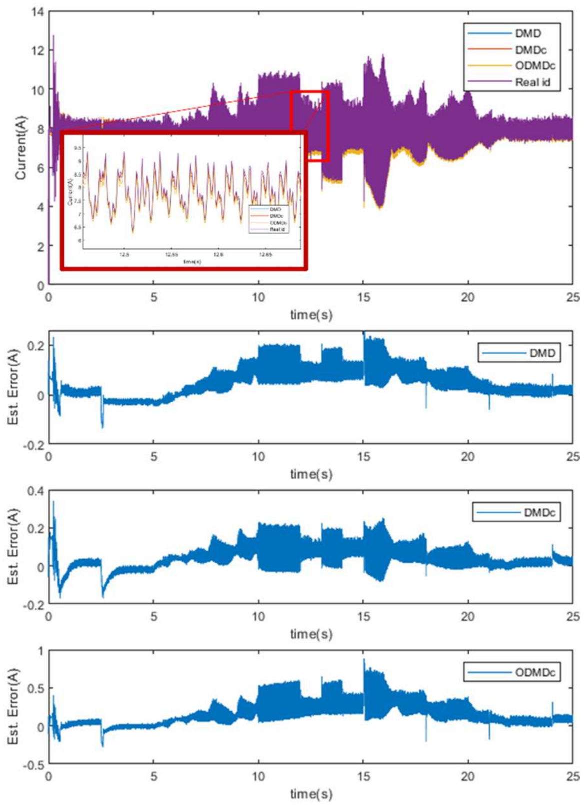


Figure 7: Estimation of i_d and residuals for three estimates.

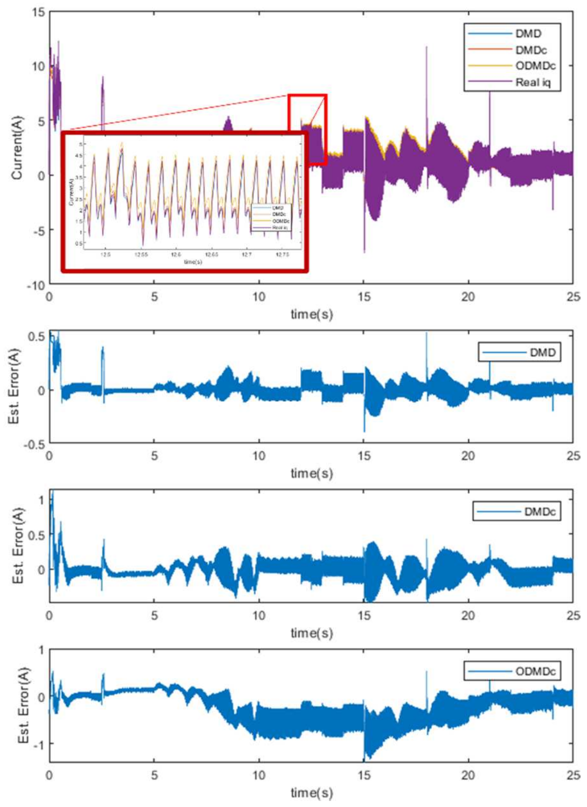


Figure 8: Estimation of i_q and residuals for three estimates.

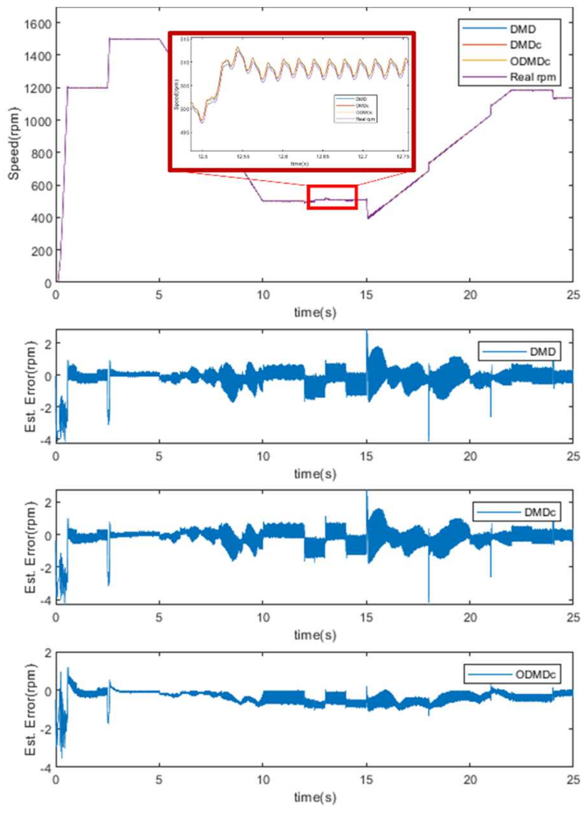


Figure 9: Speed estimation and residuals for three estimates.

Table I: Eigenvalues of ODMDc model

	λ_1	λ_2	λ_3
Initialization	0.9591	0.974+0.017i	0.974-0.017i
N=5000	0.8972	0.985+0.006i	0.985-0.006i
N=10000	0.8587	0.9887	0.9691
N=15000	0.8322	0.985+0.005i	0.985-0.005i
N=20000	0.8322	0.9923	0.9628
N=25000	0.8239	0.9925	0.9582

IV. Implementation & Experimental Results

Three algorithms are tested in the experimental setup which consisted of a VSI-fed IM coupled with a dynamometer, as shown in Figure 2. The inverter is controlled by IFOC which is built along with the estimators in dSpace.

The implementation flowchart is similar to the flowchart of simulation as depicted in Fig. 6. First a dry-run is performed to obtain machine data under varying operating conditions, then this data is used to train or initialize the algorithms. Later, these models are put into the dSpace environment for real-time estimation. Models generated by DMD and DMDc are simple state-space models following (2). However, the implementation of ODMDc is slightly different. After initialization, estimation and the model update happen sequentially in two steps. When new information (measured quantities) is received, the initialized system is updated following (10). Then, the estimation is performed following (2). These two steps are performed in every time step.

The experimental training data is shown in Fig. 10 which contains 30 seconds of data. The sampling rate is 5 kHz. The speed reference is varied between 800 and 1600 rpm while applying 1 to 2 N.m of load torque. The DMD and the DMDc models use this whole window to train, the ODMDc is initialized with 4 seconds of data. After training or initialization, models in (17)-(20) are obtained with their respective eigenvalues.

$$A_{dmd} = \begin{bmatrix} 0.967 & 0.0164 & 1.25e^{-5} \\ -0.025 & 1.0033 & 3.847e^{-5} \\ 0.709 & 0.202 & 0.9978 \end{bmatrix} \tag{17}$$

$$\lambda_{1,2,3-DMD} = [0.999, 0.9841 \pm 0.0095i] \tag{18}$$

$$A_{dmdc} = \begin{bmatrix} 0.9477 & 0.019 & 1.81e^{-4} \\ -0.046 & 1.0022 & -1.21e^{-4} \\ 2.926 & -0.9 & 0.956 \end{bmatrix} \tag{19}$$

$$\lambda_{1,2,3-DMDc} = [0.9422, 0.9819 \pm 0.011i] \tag{20}$$

These models are put into dSpace for real-time estimation. The experimental test data is shown in Fig 11. As in the simulation, ramp type input is applied with a constant 1.6 N.m load torque. The speed is varied between 800 and 1600 rpm as well. The estimation results and residuals are given in Figures 12-14 in a similar fashion to the simulated results for comparison ease. As seen from Figures 12-14, estimates follow the reference very closely with minimal error.

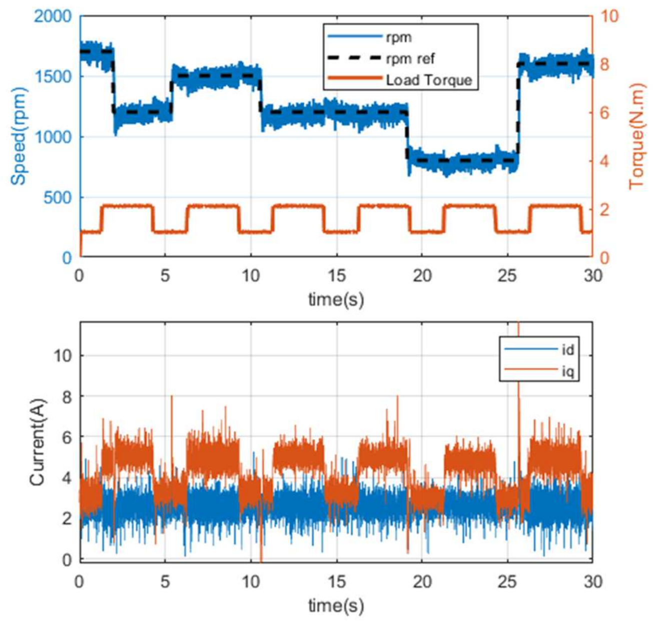


Figure 10: Experiment conditions for the training/initialization phase. Top: Speed reference, actual and load torque, bottom: id and iq.

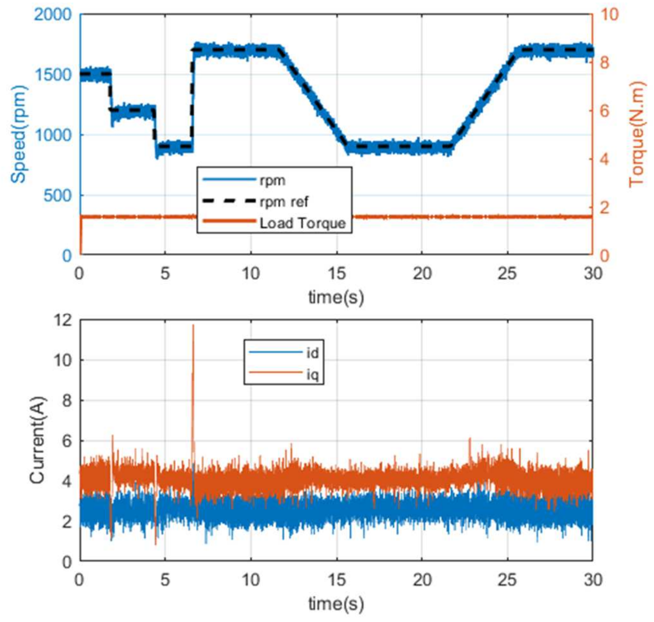


Figure 11: Experiment conditions for the testing phase. Top: Speed reference, actual and load torque, bottom: id and iq.

V. Comparison Of The Implemented Algorithms & Initialization

These algorithms are compared for performance and computational cost, and it is important to select the right parameters when comparing them. Since DMD and DMDc are considered offline and ODMDc is considered online, comparing their real-time processing time is not informative. Comparing performance is straightforward, but comparing the computational cost needs more elaboration. The whole estimation operation should be dissected into different windows: 1) Training/ initialization time, 2) Real-time estimation time, 3) Model update time. DMD and DMDc do not have the model update time. Instead of providing processing times for each processing window, the number of operations can be calculated.

For the initialization phase, there is a pseudo-inverse and matrix multiplication operation. This is valid for all three cases. For the real-time estimation phase, there are matrix multiplication and matrix addition operations. This is again common for the three algorithms. For the model update, there are additional operations as shown in (10).

Using MATLAB built-in timers, training/ initialization times are measured for the DMD, DMDc and ODMDc. They are 6.7ms, 11ms, and 2.9 ms, respectively. For real-time estimation, DMD has $n*(n-1)=:O(n^2)$ operations, DMDc and ODMDc have $n*(n-1)+m(n-1)=:O(n^2)$ operations, where n is the number of states and m is the number of control inputs. In our case n=3 and m=2. The real-time model update is unique to the ODMDc, so we cannot compare it to other estimators.

The training time will vary based on the data length, but the numbers give a perspective. Since DMDc uses both output and control input data, it takes more time to train. Whereas DMD uses outputs only. ODMDc can be initialized with only 3-4 seconds of data regardless of the total data length, so the training time will not scale up as data length gets longer.

Performance-wise, the experimental estimation errors are calculated and normalized with the maximum peak-to-peak value of the estimated state. For example, the maximum value of i_d is 5A, maximum estimation error for DMD in Fig.13 is 0.1A, which yields about a 2% maximum estimation error. Table II shows the estimation errors for the rest of the results.

Table II: Maximum estimation errors

	i_d	i_q	$speed$
DMD	2%	0.4%	0.1%
DMDc	3.4%	0.8%	1%
ODMDc	2.4%	0.8%	2%

As for the initialization of ODMDc, it uses an initial system matrix shown in equation (3), and it modifies it as new data arrives. This approach reduces the computational time as only additions and multiplications are performed for each variable, rather than performing an inversion. As good as it seems, it brings the problem of initialization. If not initialized correctly, the estimations can have large offsets or then can even go unstable. For real-time applications, initialization can be done offline and required information can be given to the algorithm manually.

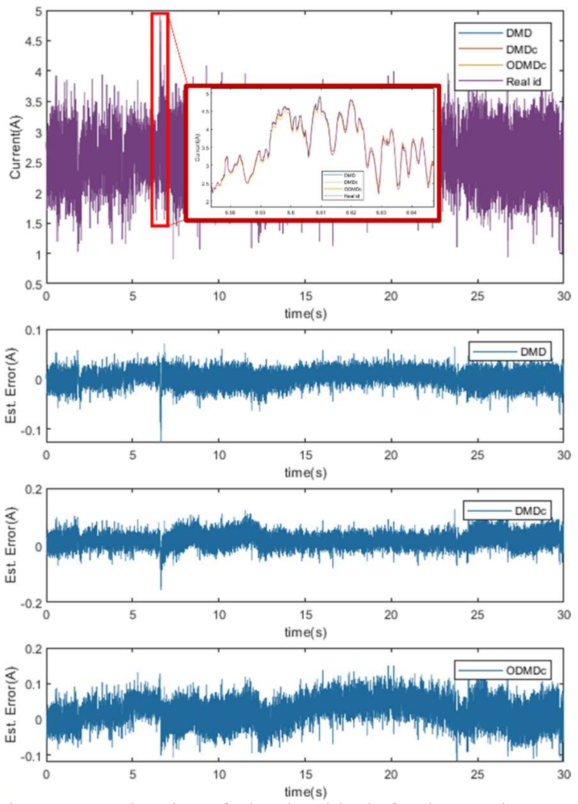


Figure 12: Estimation of *id* and residuals for three estimates.

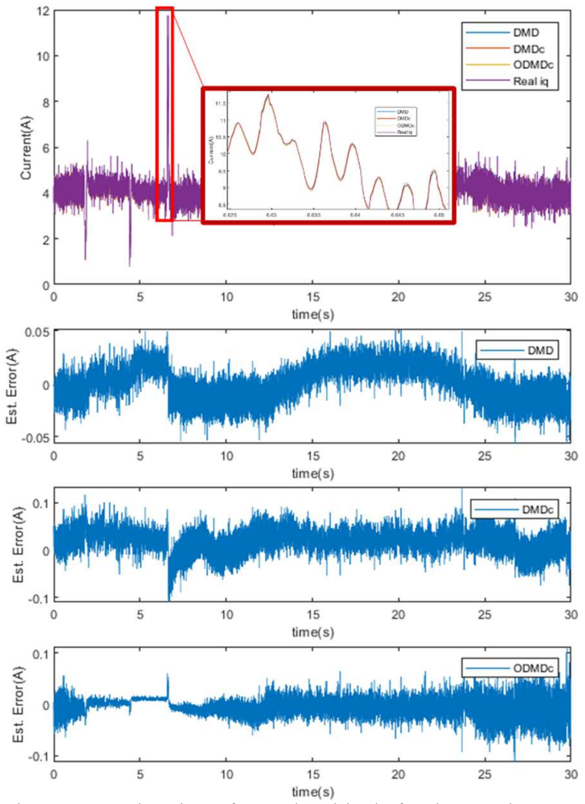


Figure 13: Estimation of *iq* and residuals for three estimates.

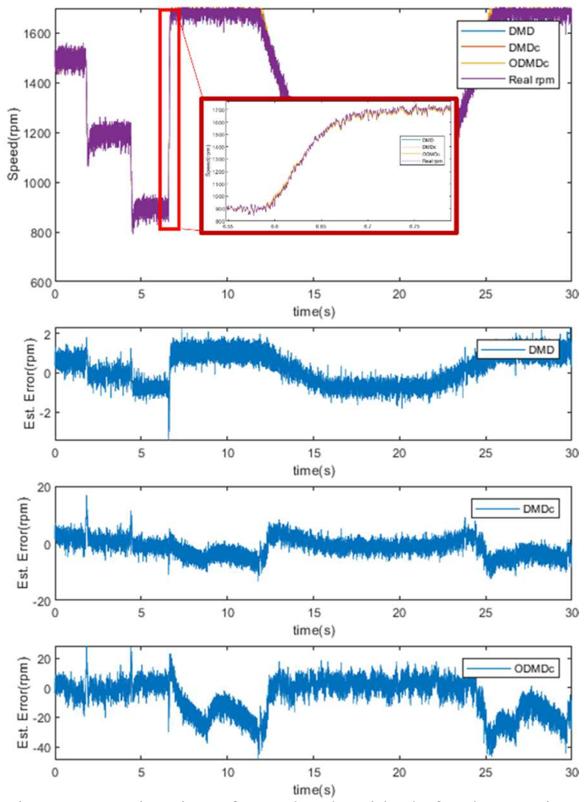


Figure 14: Estimation of *speed* and residuals for three estimates.

There are two points to consider when creating the initialization matrix. The first one is to include some portion of the transient and some portion of the steady-state behavior. But if a system is initialized in the transient state, it will not perform well. The second point is to include different excitations so that during the initialization phase, each dynamic is captured and modeled correctly.

VI. Conclusion, Potential Applications and Future Work

In this paper, dynamic mode decomposition with control and its online extensions are presented and analyzed for data-driven state estimation in IM drive systems. Derivations of these algorithms are shown to include control signals. Their performances are compared and some insight regarding initialization is given. Rotor speed and *qd*-currents are estimated. The estimation error in steady-state is satisfactory and well below 3.4% in all cases for all parameters.

Potential applications include but are not limited to health monitoring, adaptive control, system identification under faults, fault detection and isolation.

References

[1] H. Kubota, K. Matsuse and T. Nakano, "DSP-based speed adaptive flux observer of induction motor," in *IEEE Transactions on Industry Applications*, vol. 29, no. 2, pp. 344-348, March-April 1993.

[2] S. Bolognani, R. Oboe and M. Zigliotto, "Sensorless full-digital PMSM drive with EKF estimation of speed and rotor position," in *IEEE Transactions on Industrial Electronics*, vol. 46, no. 1, pp. 184-191, Feb. 1999.

[3] H. Kubota and K. Matsuse, "Speed sensorless field-oriented control of induction motor with rotor resistance adaptation," in *IEEE Transactions on Industry Applications*, vol. 30, no. 5, pp. 1219-1224, Sept.-Oct. 1994,

[4] T. Orlowska-Kowalska and M. Dybkowski, "Stator-Current-Based MRAS Estimator for a Wide Range Speed-Sensorless Induction-Motor Drive," *in IEEE Transactions on Industrial Electronics*, vol. 57, no. 4, pp. 1296-1308, April 2010

[5] H. Zhang, C. W. Rowley, E. A. Deem, and L. N. Cattafesta, "Online dynamic mode decomposition for time-varying systems," SIAM Journal on Applied Dynamical Systems, vol. 18, no. 3, pp. 1586–1609, 2019

[6] M. A. Gultekin, Z. Zhang, and A. Bazzi, "Data-driven modeling of inverter-fed induction motor drives using DMDc for faulty conditions," in 2021 IEEE International Electric Machines Drives Conference (IEMDC), 2021, pp. 1–5.

[7] J P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data, "Journal of fluid mechanics, vol. 656, pp. 5–28, 2010

[8] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," SIAM Journal on Applied Dynamical Systems, vol. 15, no. 1, pp. 142–161, 2016.

[9] P. C. Krause, O. Wasynczuk, S. D. Sudhoff and S. Pekarek, Analysis of electric machinery and drive systems, Wiley Online Library, vol. 2, 2002.