Dynamic Mode Decomposition for Real-time System Estimation of Induction Motor Drives

Muhammed Ali Gultekin, *Student Member, IEEE*, Zhe Zhang, *Member, IEEE,* Ali Bazzi, *Senior Member, IEEE*

***Abstract*— There are many methods for real-time estimation and identification for induction motor (IM) drives. In this study, dynamic mode decomposition with control (DMDc) and its variants are explored for voltage source inverter fed IM. For real-time identification, online variants of the DMDc are explored, including windowed online DMDc. These algorithms are tested in simulation for healthy and faulty behaviors of the IM drive. They are compared for estimation accuracy and implementation complexity. Lastly, these algorithms are tested on a dSpace experimental platform to further prove their potential and applicability for both healthy and faulty cases. With accurate DMDc estimation results across different variants, their applications have potentials in condition monitoring or fault-tolerant control.**

***Index Terms*—Dynamic mode decomposition, induction motor drives, real-time estimation.**

I. INTRODUCTION

Modeling the faulty behavior of any system helps in establishing strategies to mitigate faults and build model-based fault-tolerant systems. This can be achieved through control actions by estimating the next states in the system. If next states are accurately estimated, more precise and effective control signals can be generated.

Building an estimator for real-time applications can be challenging. In addition to uncertainty contributed by knowledge of the system itself and 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 like state observers, or they can be probabilistic like Kalman filters.

Other than using a model for the estimation, input and output data can be utilized to generate a model, where the model is either unknown or it has a high order. Such models are called data-driven models and they do not rely on system parameters. Many popular models generated by artificial intelligence or machine learning algorithms can be called data-driven but, may not produce an analytical or mathematical representation of the system being considered.

Dynamic mode decomposition (DMD) is such a data-driven estimation and identification technique that does not require any system information to model a system. DMD uses state measurement snapshots to model system dynamics [1]. For dynamical systems, if the control input information is available, more precise models can be generated through DMD with control (DMDc) [2].

For motor drive systems, estimation is a mature area. The main advantage of the DMDc is the nature of no-information estimation. There are established estimation techniques to estimate system states such as speed [3-5], flux [3], current [6] and position [4] or parameters such as rotor resistance [5], for control purposes. All these methods do require either machine parameters or complete machine equations to accomplish their estimation processes, which is a disadvantage due to machine parameter variations with temperature and other ambient factors which lead to parameter uncertainty.

DMD is a powerful tool that has been mostly applied to fluid dynamics. Apart from fluid dynamics research, it has applications in image processing, forecasting, and model predictive control [7-9]. DMD finds uses in power systems as a tool for oscillation analysis [10,11] and inertia estimation [12]. For control and estimation applications, DMD is coupled with a Kalman filter in [13]. Additionally, it is used as a technique of denoising in [14]. Even though it is a powerful algorithm, applications of DMD and variants is not widely explored in power electronics applications where fast dynamics exist due to switching events.

One drawback of DMDc is the demanding nature of the algorithm for real-time applications. Researchers proposed variations for the DMD for online applications [15, 16]. The derivation of online DMD (ODMD) is made in these articles yet it is not generalized if the control signal is present for the algorithm to achieve ODMDc.

After identifying a system model, making predictions based on the identified system is not a major challenge and is not a new topic in control and estimation theory. Any observer-based system can be considered as a predictor, as well as Kalman filters and neural networks. While these are viable alternatives, they require system information in terms of system matrices, as well as information regarding the environment (noise, disturbance, etc.). The power of DMD-based methods is that they are completely data-driven and can model a system based on state measurements without requiring any system parameter information. For example, in induction machine (IM) applications, DMD does not require stator or rotor inductances and winding resistances. This makes a DMD-generated model robust against parameter variations due to temperature and other factors. The performance of DMD when compared to other estimation and prediction tools is compared in [17], where DMD is shown to be superior to the Kalman filter and back-propagation neural networks in both mean absolute error and root mean squared error for all cases considered. When it comes to compare the DMD with similar tools such as proper orthogonal decomposition [18], DMD has a faster response, but it falls behind in accuracy. Since this paper focuses on real-time systems, DMD is of more interest due to its faster response.

One of the uses of real-time system identification and estimation is condition monitoring. The estimated quantity can be frequency, a system parameter such as DC link capacitance or a system state such as rotor flux [19-22]. Though very promising and important topics, this is beyond the scope of this paper.

The main purpose of this paper is to investigate DMDc variants that run online or in real-time under healthy and faulty conditions by illustrating the estimation process, and guide future researchers who wants to use different DMDc variants. Achieving accurate estimation and state prediction results during both healthy and faulty conditions can significantly contribute to fault-tolerant system development. The presented DMDc variants are implemented in MATLAB and applied to an IM drive application. Simulations are made for healthy and faulty operating conditions. The algorithms are also experimentally tested on a lab prototype where they are implemented on a dSpace platform. Healthy and faulty conditions are tested. Comparisons among the DMDc variations are made using simulation and experimental results.

The paper is organized as follows: In Section II, the DMDc algorithm and mathematical background are given along with derivation of windowed ODMDc (WODMDc). In Section III, simulation results are presented for the induction motor drive system to demonstrate how DMDc variants accurately estimate system states under healthy and faulty conditions. Section IV presents the implementation considerations and comparison among DMDc algorithms. Hardware test scenarios for healthy and faulty operation results are given in section V. Section VI concludes the paper.

II. Background on DMDc and its Online Variants

Consider a discrete-time dynamical system with a system matrix A, an input matrix B, a state vector $x[k]$ and an input vector $u[k]$. It is assumed that the system model does not change in between sampling intervals and it is linear. Since the modeling technique we adopt (DMD) is a linear one, even though the system is non-linear, the generated model will be linear. The vector $x[k]$ is constructed from measurable quantities; that is, it contains only measurable states and not necessarily states available in a standard dynamical model. The goal is to estimate system and input matrices and use them to predict future states using available measurable states. Although the generated models will not be identical to the dynamical model, it is expected from these models to show similar behaviors under similar operating conditions. For example, a typical squirrel-cage induction machine dynamical model may include rotor quantities that are not measurable; for DMD purposes, only measured voltages, currents and speed would be used to construct $x[k]$. Section V elaborates on $x[k]$ and $u[k]$ in DMD applications.

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

If we consider $u[n]$=0 for a non-excited system, we will have:

$$x[k+1] = A\,x[k] \tag{2}$$

$$A \;=\; x[k+1]\,(x[k])^{-1} \tag{3}$$

The system matrix can be found from (3) and we can re-write (3) as:

$$A \;=\; x[k]\,(x[k-1])^{-1} \tag{4}$$

Using the measured input and output data, the system is modeled, and system matrices A and B are generated. Using generated system matrices, the next state is synthesized. The algorithm describing this process is given in Fig.1.

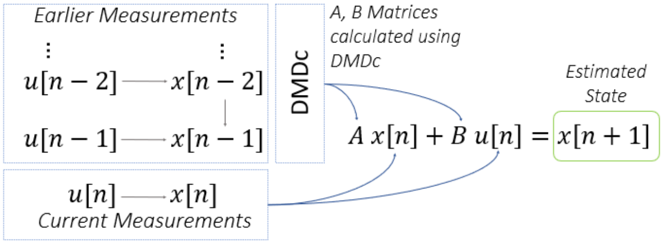

Fig.1: Predictive modeling process.

An interpretation of (4) is that if we have state measurements $x[k]$ and a previous sample of the state measurements $x[k-1]$, we can find matrix A. So far, equations (1)-(4) summarize the DMD algorithm. In the regular DMD algorithm, an order reduction step follows (4), but for the system of interest, which is an IM drive, it is not necessary since the order is already low. To improve this method for feedback systems, control signals can be added to (4) to form DMDc. To solve the system given in (1) with the same methodology, new matrices will be defined. Let,

$$G = [A, B], \qquad \theta_n = [X_n; U_n] \tag{5}$$

$$X_n = [x_1, x_2, \dots, x_n], \qquad U_n = [u_1, u_2, \dots u_n] \tag{6}$$

where X_n is the collection of n samples of the state measurements and U_n is the collection of n samples of the input measurements as shown in (6). From these definitions, it is easy to follow,

$$X_{n+1} = G\,\theta_n \tag{7}$$

$$G = X_{n+1}\,(\theta_n)^{-1} \tag{8}$$

$$G = X_n\,(\theta_{n-1})^{-1} \tag{9}$$

The matrix θ has the measurement for the last n samples, $X_{n+1}$is the last n samples of state measurements, shifted by one. θ_n is most probably a non-square and thus non-invertible matrix. In such a case, taking pseudo-inverse utilizing singular value decomposition (SVD) should be considered. Equation (9) can be re-written considering pseudo-inverse (shown as †):

$$G = X_n\,(\theta_{n-1})^{\dagger} \tag{10}$$

Where the pseudo-inverse is defined as:

$$\Theta_k^{\dagger} = \Theta_k^{T}(\Theta_k\,\Theta_k^{T})^{-1} \tag{11}$$

It should be noted that for the models we are interested in, equation (10) poses an over-defined system where the order of the system is significantly less than the number of equations or measurements. So only a single G matrix can be defined.

Normally DMDc uses order reduction while using SVD, however, for low-order systems this is not necessary. The system of interest here is an inverter-fed induction motor which can be considered as low order.

This article has been accepted for publication in IEEE Transactions on Industry Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIA.2022.3231843

Fig.2 gives a perspective on how these matrices are formed for an application. To apply DMDc, the following matrices are needed: State snapshots *X*, input snapshots *U*, and augmented control matrix *θ*. By using these three matrices, the matrix *G* containing *A* and *B* will be found. X_1 is defined to be the sub-matrix of *X* with the first component excluded. Similarly, X_2 and U_2 are sub-matrices of *X* and *U*, respectively, where the last components are excluded. To form *θ*, U_2 is used. Fig. 2 illustrates such formations.

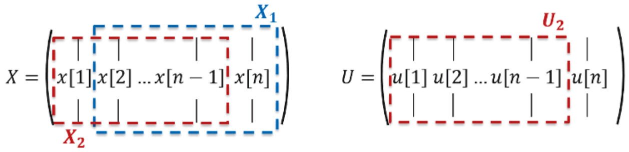

Fig. 2: Formation of matrices.

Based on Fig.2, (10) can be expressed as:

$$\theta = [X_2\,; U_2], \qquad G = X_1\theta^\dagger \tag{12}$$

The model of interest includes both the controller and the system models as seen in Fig. 3; the system dynamics might be suppressed by the controllers if the controllers' time constants are high. By referring to Fig. 3, since the DMDc and its variants are input-output type (they are functions of control signal *u* and output *y*), any feedback within the system model will be masked by the slowest dynamic (controller, actuator, or system). However, the model generated by DMDc algorithms includes a *B* matrix which map the control signals to the system states. It gives a degree of de-coupling of the controller from the system, but it is not a complete de-coupling since controllers also have dynamics.

Identifying a system using this method is useful but it is not a scalable method to run in real time. The calculation time of the *G* matrix increases as new data points come in i.e., more samples mean larger *X* and *U* matrices. That is why more practical methods for real-time or online implementation are proposed.

In classical DMDc, all data is used to calculate the *G* matrix. A more realistic approach is to use a sliding window and use a constant number of snapshots to get *G*. This method is called windowed-DMDc (WDMDc). While easy to implement, it has disadvantages that will be discussed in later sections. In WDMDc, the *G* matrix is calculated in each time sample. Even though this solves the scalability issue, it does not necessarily reduce the calculation time. Moreover, the model (*G*) will be correct only for the conditions of windowed data. For offline implementations this is not an issue since the window size can be set to represent a large amount of data and the generated model will be a general model. But for real-time implementations with a limit to the window size, a 'general' model cannot be generated. Therefore, WDMDc might have inaccurate estimates during transients. Therefore, it was proposed in [20] to implement the DMDc algorithm recursively using the least-squares estimate to achieve the online DMDc (ODMDc) and reduce computation time.

A. Online DMDc

The derivation of ODMDc is similar to the derivation of ODMD. The update formula can be generated by replacing matrix *A* with the augmented system matrix *G*, and state measurement vector *x* as *θ* shown in (5). This yields,

$$G_{k+1} = G_k + \gamma_{k+1}(y_{k+1} - G_k\theta_{k+1})\theta_{k+1}^T P_k \tag{13}$$

where *γ* and *P* are intermediate variables [20]. The ODMDc constantly updates the original system matrix. It is computationally efficient, and it does not require storing measurements in the memory. But, if the system under consideration is time-variant, relying on older measurements will cause modeling errors. This has led to Windowed-ODMDc (WODMDc), which is like WDMDc where it only considers a window of measurements but is also online [20].

B. Windowed ODMDc Derivation

Define the following where *G* is the system matrix, *θ* is the measurement vector containing a single measurement set, and Θ_k is the measurement matrix containing *k* measurement sets.

$$G = [A\ B], \qquad \theta = [x; u], \qquad x_{k+1} = y = G\,\theta \tag{14}$$

The matrices shown in Fig. 1 can be re-named as

$$Y_k = [x_2 \dots x_{k+1}\,], \qquad \Theta_k = [x_1 \dots x_k; u_1 \dots u_k] \tag{15}$$

where subscript *k* denotes the *k*th sample. The objective function J_k at the *k*th sample for the least-squares estimate is formed as,

$$J_k = \left|\left|Y_k - G_k\Theta_k\right|\right|^2, \qquad G_k = Y_k\Theta_k^\dagger\,. \tag{16}$$

The pseudo-inverse is defined previously in (11). If (11) is substituted in (16),

$$G_k = Y_k\Theta_k^\dagger = Y_k\Theta_k^T(\Theta_k\ \Theta_k^T)^{-1} = Q_k P_k \tag{17}$$

where *Q* and *P* matrices are defined as follows where *w* is the window size:

$$Q_k = Y_k\Theta_k^T = \sum_{i=k-w+1}^{k} y_i\theta_i^T \tag{18}$$

$$P_k = (\Theta_k\ \Theta_k^T)^{-1} = \left(\sum_{i=k-w+1}^{k} \theta_i\theta_i^T\right)^{-1}. \tag{19}$$

The new measurement data at (*k+1*) will alter the *Q* and *P* matrices:

$$Q_{k+1} = Y_{k+1}\Theta_{k+1}^T$$

$$= \sum_{i=k-w+2}^{k+1} y_i\theta_i^T = Q_k - y_{k-w+1}\theta_{k-w+1} + y_{k+1}\theta_{k+1}^T \tag{20}$$

$$P_{k+1}^{-1} = \Theta_{k+1}\ \Theta_{k+1}^T$$

$$= \sum_{i=k-w+2}^{k+1} \theta_i\theta_i^T = P_k^{-1} - \theta_{k-w+1}\theta_{k-w+1} + \theta_{k+1}\theta_{k+1}^T \tag{21}$$

The following definitions can help for a clearer representation:
$U = [\theta_{k-w+1}\ \ \theta_{k+1}], V = [y_{k-w+1}\ y_{k+1}], C = [-1\ 0; 0\ 1]$ (22)

This article has been accepted for publication in IEEE Transactions on Industry Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIA.2022.3231843

So, the Q and P matrices are,

$$Q_{k+1} = Q_k + V\ C\ U^T \tag{23}$$

$$P_{k+1}^{-1} = P_k^{-1} + U\ C\ U^T \tag{24}$$

The updated G matrix can be calculated by:

$$G_{k+1} = Q_{k+1}P_{k+1} = (Q_k + V\ C\ U^T)(P_k^{-1} + U\ C\ U^T)^{-1} \tag{25}$$

Using matrix inversion lemma, we can represent P_{k+1} as

$$P_{k+1} = P_k - P_k U\ \Gamma_{k+1} U^T P_k\ , \Gamma_{k+1} = C^{-1} + (U^T P_k U)^{-1} \tag{26}$$

G_{k+1} becomes

$$G_{k+1} = (Q_k + V\ C\ U^T)(P_k - P_k U\ \Gamma_{k+1} U^T P_k)$$

$$= Q_k P_k - Q_k P_k U \Gamma_{k+1} U^T P_k + V\ C\ U^T P_k - VCU^T P_k U\ \Gamma_{k+1} U^T P_k \tag{27}$$

We can write

$$\begin{aligned} V\ C\ U^T P_k - VCU^T P_k U\ \Gamma_{k+1} U^T P_k \\ = VC(\Gamma_{k+1}^{-1} - U^T P_k U)\Gamma_{k+1} U^T P_k \\ = VCC^{-1}\Gamma_{k+1} U^T P_k = V\Gamma_{k+1} U^T P_k \end{aligned} \tag{28}$$

Using Γ_{k+1} definition

$$\begin{aligned} G_{k+1} &= Q_k P_k - Q_k P_k U \Gamma_{k+1} U^T P_k + V \Gamma_{k+1} U^T P_k \\ &= G_k - G_k U \Gamma_{k+1} U^T P_k + V \Gamma_{k+1} U^T P_k \end{aligned} \tag{29}$$

Then the update equation can be stated as,

$$G_{k+1} = G_k + (V - G_k U)\Gamma_{k+1} U^T P_k \tag{30}$$

This way the effect of the last element of the measurement window can be subtracted and the contribution from the new measurement can be added without requiring recalculating the G matrix from scratch.

C. Weighted-Online DMDc (We-ODMDc)

Windowed ODMDc removes the effects of the earlier measurements and gives equal weight to all measurements within the window. It is possible to alter the update formula given in (30) to incorporate a weighing factor as shown in (31). With this addition, it becomes possible to put more weight on the latter measurements and gradually forget older measurements. The update formula for the weighted-online DMDc has the parameter ρ which is the forgetting factor. If it is wanted to have a half-life of 3 seconds, then ρ can be selected as $\rho = 2^{-1/3 * f_{sampling}}$.

$$G_{k+1} = G_k + \rho(V - G_k U)\Gamma_{k+1} U^T P_k \tag{31}$$

We-ODMDc converges to WODMDc as the half-life becomes larger. The term ρ, becomes very close to one if the half-life is selected too large.

III. VSI-Fed Induction Motor Simulations Under Nominal and Faulty Conditions

A voltage source inverter (VSI)-fed IM is considered with indirect field-oriented control (IFOC). The inverter is operated using sinusoidal PWM. The IM is modeled with the flux-based equations given in [23]. IFOC is implemented as PI-based [24]. The parameters used for the IM are provided in Table I and the system block diagram is shown in Fig.3.

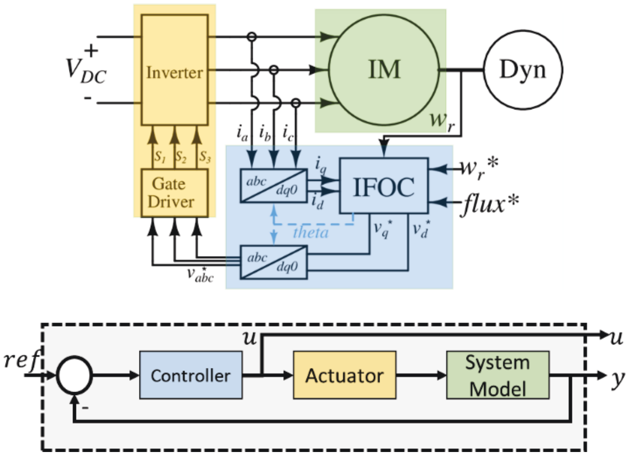

Fig.3: Block diagram for VSI fed IM and DMDc.

TABLE I: IM Parameters

$r_s(\Omega)$	1.44	$X_{ls}(\Omega)$	2.88
$r_r'(\Omega)$	1.3	$X_{lr}'(\Omega)$	4.32
$X_m(\Omega)$	83.42	$f_{base}(Hz)$	60

A. Case I: Nominal Operation with Reference and Load Change

With the nominal operation, the speed reference is changed from 1500 rpm to 1700 rpm, then reduced to 900 rpm at t=2.5s and 4s respectively. During the speed stepping, the load is also changed from 1 N.m to 4 N.m then to 3 N.m and to 6 N.m at t=1.5s, t=3 s and t=5s, respectively. The sampling time in Simulink is 10µs and the simulation is run for a total of 6s. The inverter switching frequency is 10kHz and the DC bus voltage is 300V. The plots for speed, speed reference, load torque, q-d currents and current references are given in Fig. 4. To check if IFOC is performing as expected, the regulation of q-d currents is examined.

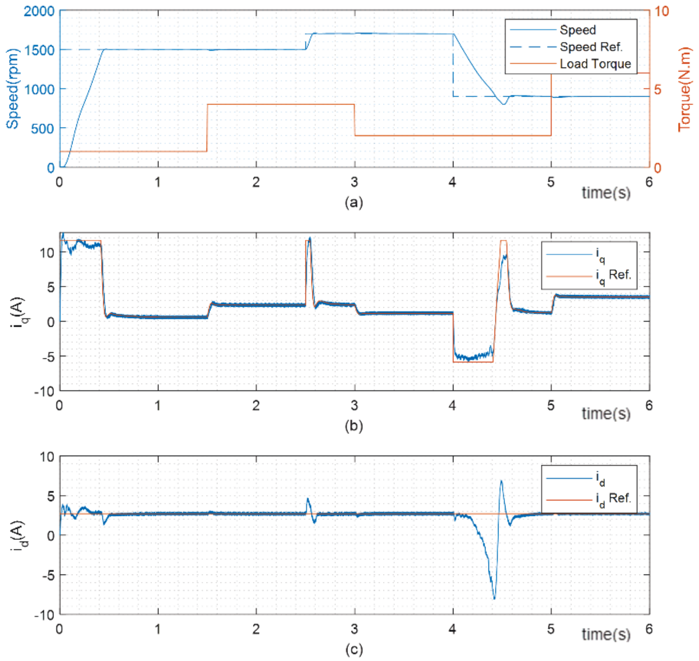

Fig.4: Simulation results for case I, nominal operation with load stepping, (a) speed and torque, (b) & (c) i_d and i_q with their references respectively.

B. Case II: System Model Change – Hardware Failure

A hardware fault is injected to the system, namely a shaft misalignment [25-27]. The shaft misalignment fault occurs in the bearings of a machine, and it varies the airgap. Varying airgap modifies generation of flux within the machine. Fig. 5 illustrates the shaft misalignment and varying airgaps. The misalignment of the rotating center modifies the generated flux. In effect, this flux will be reflected in the stator currents as added harmonics following (32) [25].

$$f_{ecc} = f_e\left[1 \pm \left(\frac{1-s}{P/2}\right)\right] \qquad (32)$$

where f_e, s and P represents the input frequency, the slip and the number of poles respectively. For a four-pole machine, (32) reduces to

$$f_{ecc} = f_e\left[1 \pm \left(\frac{1-s}{2}\right)\right] = f_e \pm f_r \qquad (33)$$

where f_r represents the mechanical rotation speed. The added harmonics are centered around the input frequency and vary by the rotational speed.

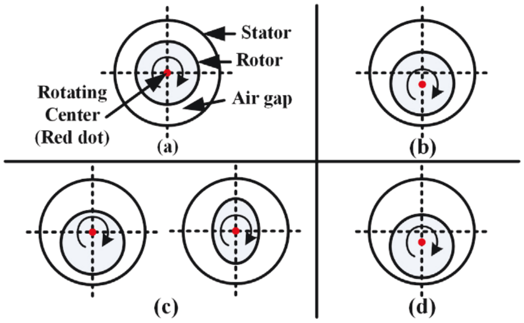

Fig. 5: Different types of shaft misalignments and varying airgaps [26]

The shaft misalignment is a complex fault to model. It changes the airgap and inductances of each phase dynamically. In this paper, only effects of the fault are modelled rather than the fault mechanism itself which is side-band frequency harmonics in the current. For more accurate and complete models for the eccentricity fault please refer to [28, 29]. The addition of extra harmonics causes ripples in the fluxes and currents, yet the speed is regulated well. Since the model used in this article has fluxes and currents as state variables, they are readily available in the Simulink model. A block diagram is shown in Fig. 6. By injecting two harmonics to the stator current (i_{abc}), flux variation effect is achieved. In healthy simulations the fault injection block is by-passed.

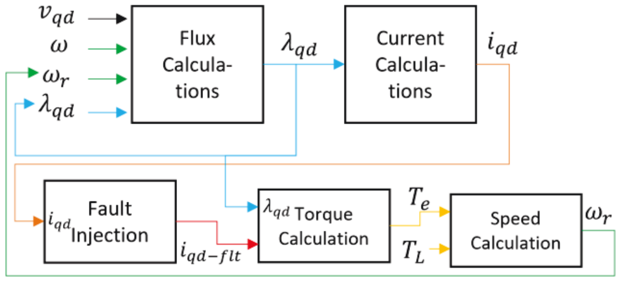

Fig. 6: IM model, eccentricity fault injection block diagram

The simulation results for this case are given in Fig. 7. The speed reference is kept constant at 1500 rpm with a constant load of 1 N.m. The fault is injected at t=2s where the total simulation time is set to 4 seconds.

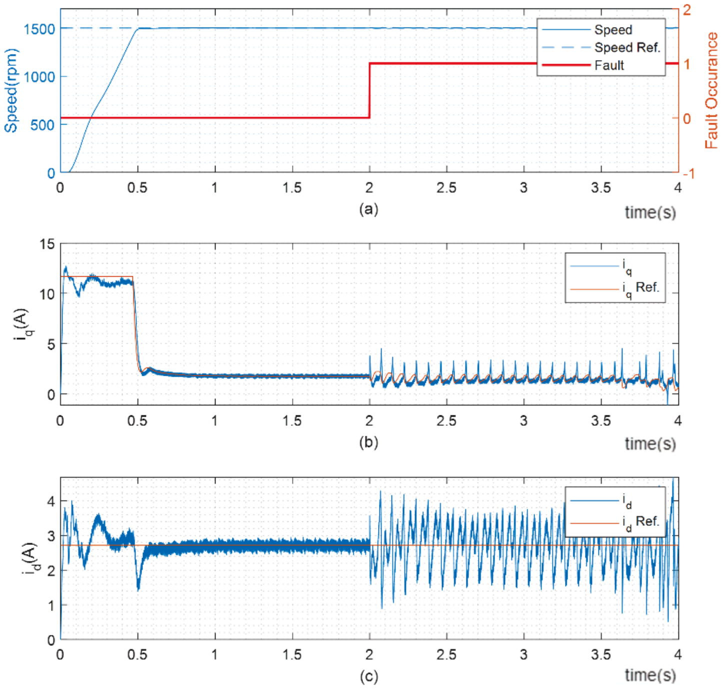

Fig.7: Simulation results for case II, hardware fault, (a) speed and fault occurrence signal, (b) & (c) i_d and i_q with their references respectively.

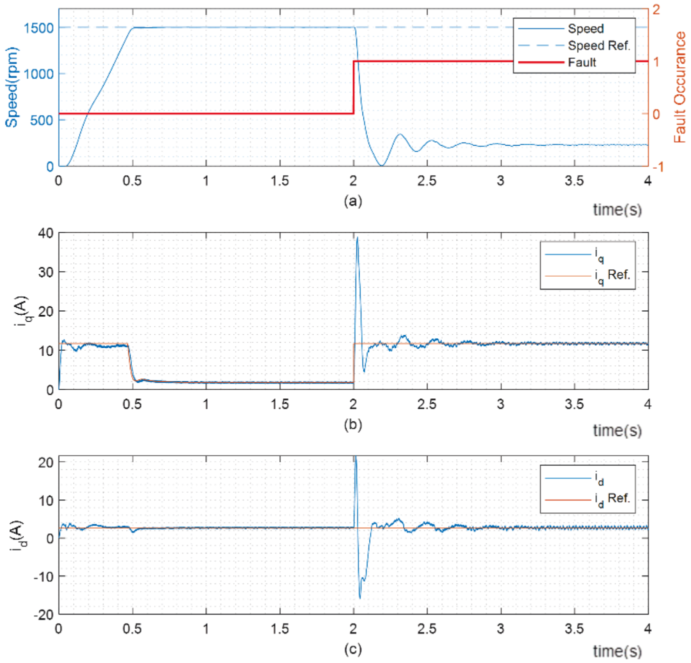

Fig.8: Simulation results for case III, speed sensor failure, (a) speed and fault occurrence signal, (b) & (c) i_d and i_q with their references respectively.

C. Case III: Speed Sensor Failure

The third case is the speed sensor failure where the speed sensor feedback is omitted. This fault is injected at t=2s where total simulation time is 4 seconds. The system response changes suddenly under this failure as shown in Fig. 8.

IV. Software Implementation, considerations and algorithm Comparison of DMDc Variants

A. Software Implementation

For the system shown in Fig. 3, we define $x[k]$ and $u[k]$ as shown in (34) and (35). These use available measurable states or quantities rather than the classical definition of an IM drive model.

$$x[k] = \left[i_q[k], i_d[k], w_r[k]\right]^T \tag{34}$$

$$u[k] = \left[v_q^*[k], v_d^*[k]\right]^T \tag{35}$$

All four algorithms are applied to the three test cases mentioned in Section III. *It is important to note that when estimation is discussed, it also includes one-step-ahead prediction where x[k+1] is the estimated quantity*. Estimation results for these algorithms are shown Figures 9, 10, and 11 for i_q, i_d, and ω_m, respectively, for case I. Residuals for cases II and III are not shown here to make plots more readable, but their probability distributions are shared later in this section. Residuals can be compared to determine which algorithm performs best for each case and estimated state. Residuals are calculated using (36) where res_k is the residual at sample k, and the "^" symbol represents an estimated signal.

$$res_{k+1} = x_{k+1} - \hat{x}_{k+1} \tag{36}$$

As clear in Figures 9-11, the algorithms' estimation performances are quite similar for the steady state. They differ in transient response and settling times. Plots start after initializations are done.

B. Considerations

1) Window size, shifting index selection and rate transitioning for WDMDc and WODMDc

One important criterion for windowed versions is the selection of window size. Since in each sampling-time there is a calculation stage, the duration of this stage is critical for real-time implementation. Other than calculation time, data storage capability is another concern for windowed versions. Every sample within the window should be stored. This creates a tight margin for the selection of window size if memory is limited. Moreover, as the system order increases, this criterion becomes tighter.

The shifting index and rate transition are very similar parameters where they decide how much down sampling needs to be done. The difference lies in the implementation detail, the rate transition reduces the calculation cost associated with data logging whereas shifting index reduces actual calculation cost. The shifting index and rate transitioning are inversely proportional to the computational burden. They can be useful features to overcome hardware restrictions.

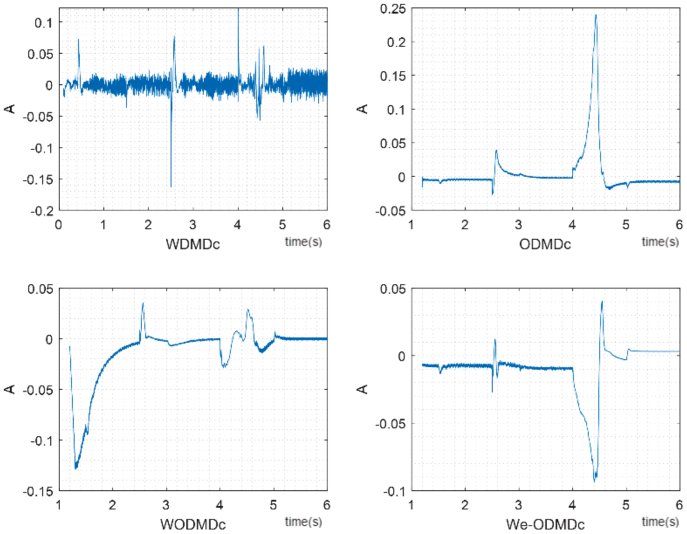

Fig. 9: i_q residuals for the case I,

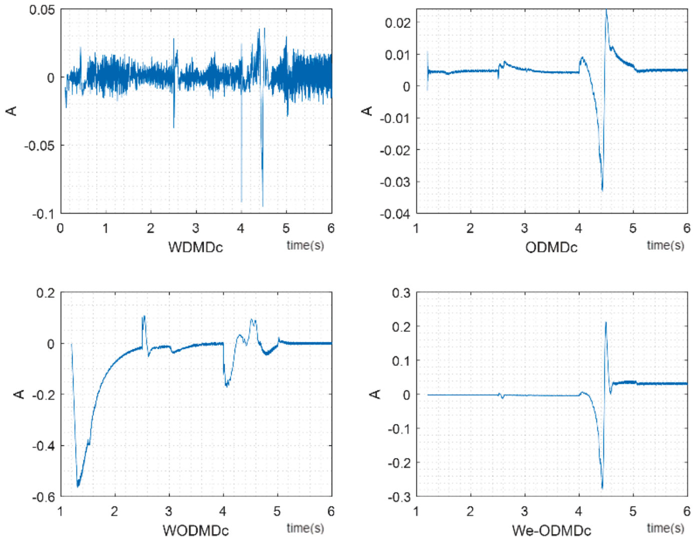

Fig. 10: i_d residuals for the case I,

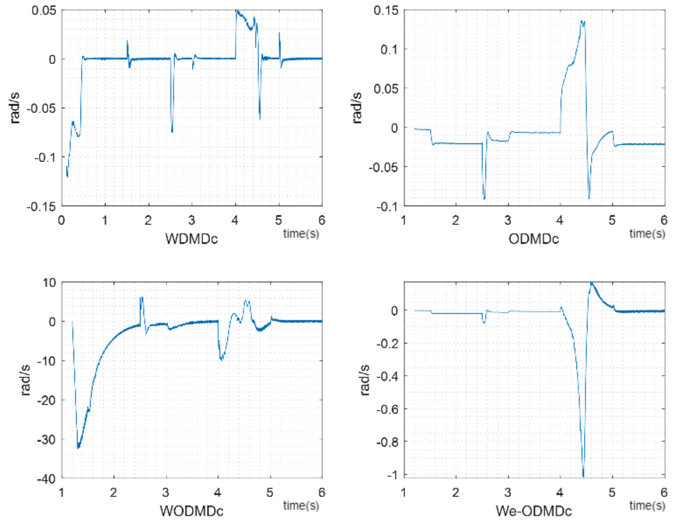

Fig. 11: ω_m residuals for the case I

2) Initialization of ODMD, ODMDc, WODMDc

Online implementations of DMDc use an initial system matrix G_0 when k=0 in (30) and they modify the initialization matrix as new data comes in. This approach reduces the computational burden as only additions and multiplications are performed for each variable, rather than performing an inversion.

However, this highlights the initialization problem where estimations can have large offsets or diverge if not initialized correctly. For real-time applications, initialization can be done offline and required information can be given to the algorithm manually. There are two points to consider when creating the initialization matrix: 1) include some portion of the transient and some portion of the steady-state behavior to capture both, and 2) include different excitations and operating conditions so that the initialization phase can capture various state transients. For an induction motor, startup from zero speed to commanded speed should be included with multiple values of speed commands and load torques to ensure the validity of the initialized system. If only steady-state data is included in the initialization phase, the initial model will not reflect transients and the estimations will have either large offsets or they won't diverge.

An example of the initialization problem is given in Fig. 12. The experimental data presented in the paper is used and ODMDc is used as a case study. The 'Estimator 1' is initialized from the first two seconds as an example of "a bad initialization" and the 'Estimator 2' is initialized from the first four seconds as an example of a "good initialization". The second case includes a transient response but the first case does not. It can be observed from Fig. 12 that the first case has a large estimation error in the first transient it sees at t=2.5s. Whereas the second estimator successfully estimates the transient at t=6.6s.

One important aspect of the DMDc variants is the continuous update of the model. For example, in Fig. 12, even though the first case had a large swing in the first transient, it captures the second transient better, albeit with an offset. As machine operation continues, the model converges to the real model. To better reflect the transients, all the measurements and estimations are filtered with a window size of 50 samples for the example in Fig. 12.

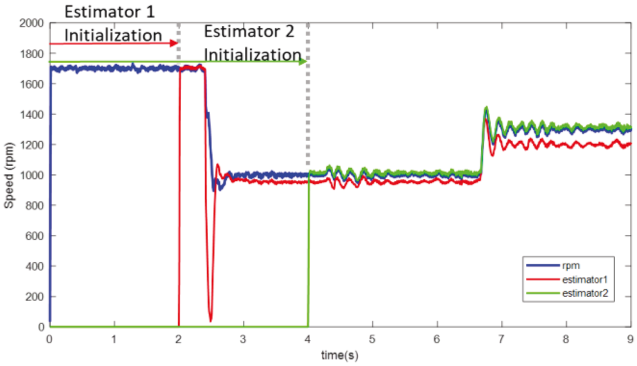

Fig. 12: The good and bad initialization example. Blue line is the actual speed of the machine, red and green lines are estimator 1 and 2, respectively.

C. Comparison

To make a clear comparison, the same parameters are used for these four algorithms. These parameters are, the window size, the shifting index, the initialization window, forget factor and they are chosen as 500 samples, 1, 6000 samples and 1 s, respectively.

Estimation is accurate under healthy or nominal operating conditions as simulated in case I. Residuals from cases II and III follow similar patterns to case I. To illustrate this, the probability densities of the residuals from all three cases are plotted. The densities are centered around or close to zero, indicating the estimation errors are low under both healthy and faulty conditions. It is important to note that in cases II and III, half of the data is under healthy conditions and the other half under faulty conditions, which explains multiple peaks in some plots in Figures 13-15.

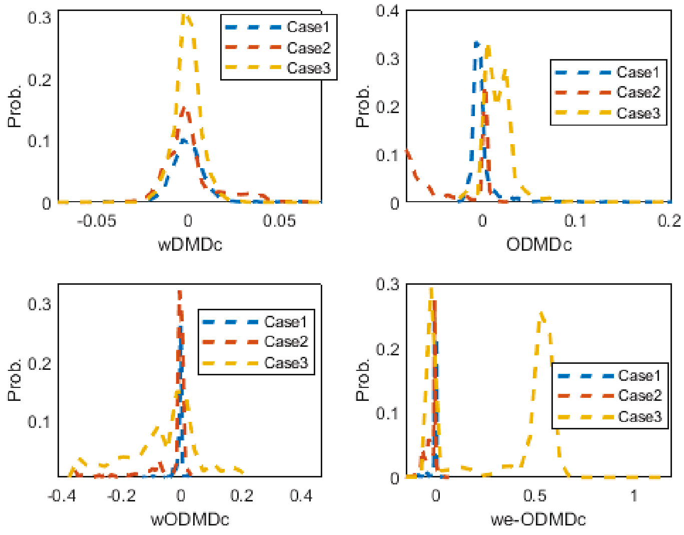

Fig. 13: Residual probability distributions for i_q

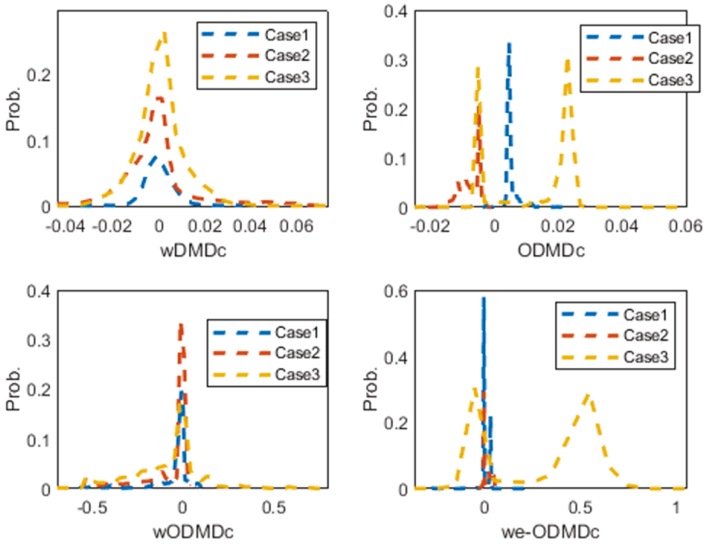

Fig. 14: Residual probability distributions for i_d

Even though the execution time is not a very accurate metric to compare these algorithms, it gives an idea about the computational burden for the same length datasets. The experimental dataset is run offline in MATLAB and the

This article has been accepted for publication in IEEE Transactions on Industry Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIA.2022.3231843

timings are measured using built-in timers in MATLAB. Results are given in Table II, and as expected, WDMDc takes much longer to process than online variations.

TABLE II: EXECUTION TIMES

WDMDc	ODMDc	WODMDc	We-ODMDc
1.8s	0.038s	0.142s	0.043s

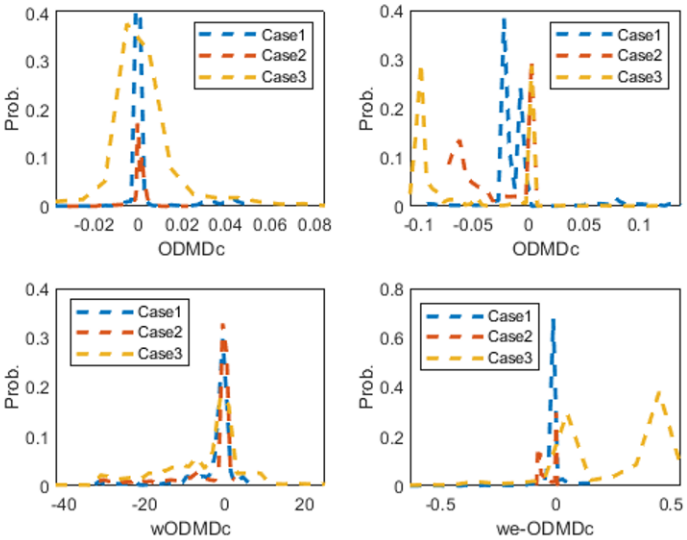

Fig. 15: Residual probability distributions for w_m

V. Experimental Results

The experimental platform is a VSI-fed IM setup shown in Fig. 16. The IM is connected to a dynamometer to measure or control speed or torque. The induction motor is a four-pole, 1.5 hp, 208V squirrel cage motor. The inverter is controlled by dSpace DS1104, employing SPWM switching patterns. The gate signals are generated by dSpace using the IFOC algorithm. The sampling frequency (f_{samp}) and switching frequency (f_{sw}) are different for each algorithm as f_{samp} dictates the time window for computations to happen. As f_{samp} is varied, f_{sw} is also changed to have consistent controller operation. Two test scenarios are considered: Healthy case and an inverter failure.

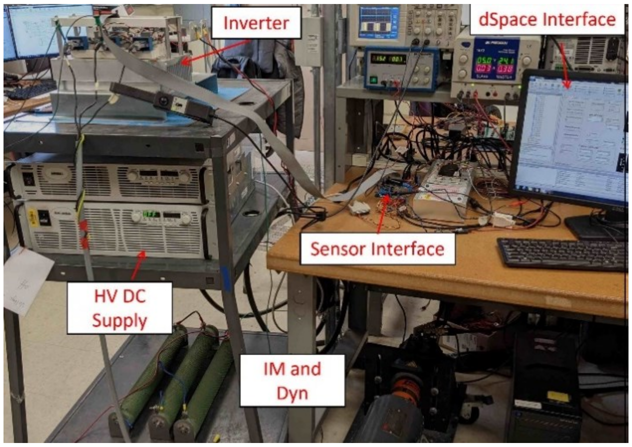

Fig 16: Hardware setup consisting of a DC supply, an inverter, an IM coupled with a dynamometer and dSpace interface.

A. Scenario 1: Healthy Case

During the healthy case, a speed and load pattern shown in Fig.17(a) is used. The speed is stepped from 1300 rpm to 1700rpm, and it is reduced to 1000 rpm. During these transitions, load torque is stepped from 1N.m to 3N.m and down to 1N.m again. Currents i_d and i_q are following the references closely in Fig. 17 (b) and (c) which indicates valid FOC operation.

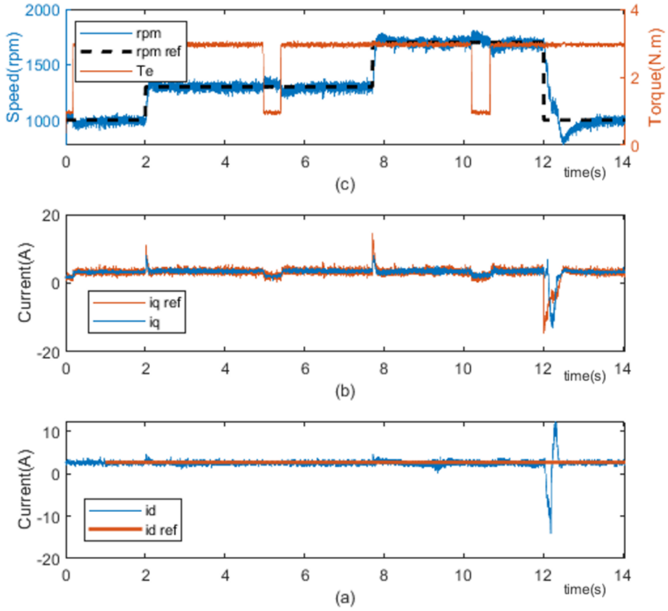

Fig.17: Experiment conditions, (a) speed reference, speed measurement and load torque, (b) and (c) i_q and i_d references and measured, respectively.

The first implemented and least burdensome algorithms are ODMDc and We-ODMDc. They do not require data storage and their computational costs are lower. When implementing, the sampling and switching frequencies are selected as 5kHz and 2.5kHz, respectively. The half-life for the We-ODMDc is selected as 5 seconds. For initialization, the system runs without estimations, data is collected for 10 seconds, and those measurements are used to calculate initial G and P matrices as given in equations (17) and (19).

Secondly, WODMDc is implemented, which requires data storage to update properly. The initialization is done similarly to the previous two models but $f_{sampling}$ and f_{sw} are changed to 4kHz and 2kHz respectively. To have better estimation accuracy and longer window size, the signals are down sampled by a factor of 20. That means the sampling time for the motor control is 4kHz, but it is 200Hz for the WODMDc. In this configuration, a window of one second is achieved.

Lastly, the WDMDc is implemented which is the most demanding algorithm. $f_{sampling}$ and f_{sw} are changed to 3.2kHz and 1.6kHz respectively, with down sampling at a factor of 16 to reach one second window.

The estimation residuals are calculated using equation (36) and they are shown in Figures 18-20. The first noticeable observation is the variations in the experimental residuals are higher than the simulation data due to noise apparent in the measurement system. To examine the distribution more closely the probability densities of residual distributions are

This article has been accepted for publication in IEEE Transactions on Industry Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIA.2022.3231843

plotted for the experimental data in Fig. 21. There are offset errors in online variations where WDMDc does not have any. The offset errors are slightly rectified in windowed and weighted ODMDc when compared to regular ODMDc. It can be observed from these figures that the overall estimations are correct. It can also be inferred from the residuals that the system correctly represents initialization data, so that under changing conditions or with new data, estimates still converge correctly.

The variance of WDMDc is the largest one even though it does not have any offsets. It is a matter of parameter tuning as explained in [30], by selecting a larger window size or increasing down-sampling rate, the variation can be minimized.

B. Scenario 2: Faulty Case

To show the strength of the estimations, a fault scenario is designed. For the fault, open circuit failure is considered as one of the inverter phases is set to zero current. This is emulated by disconnecting one phase from inverter to the IM. The phase current waveforms at the failure are shown in Fig. 22.

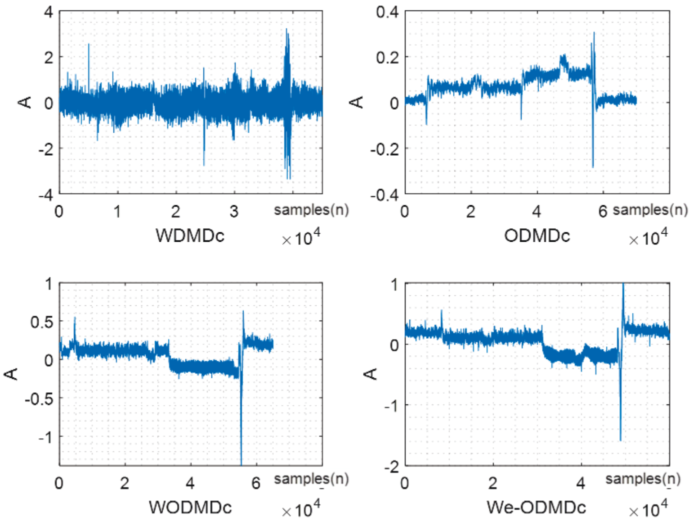

Fig.18: i_q residuals for the healthy experimental implementation.

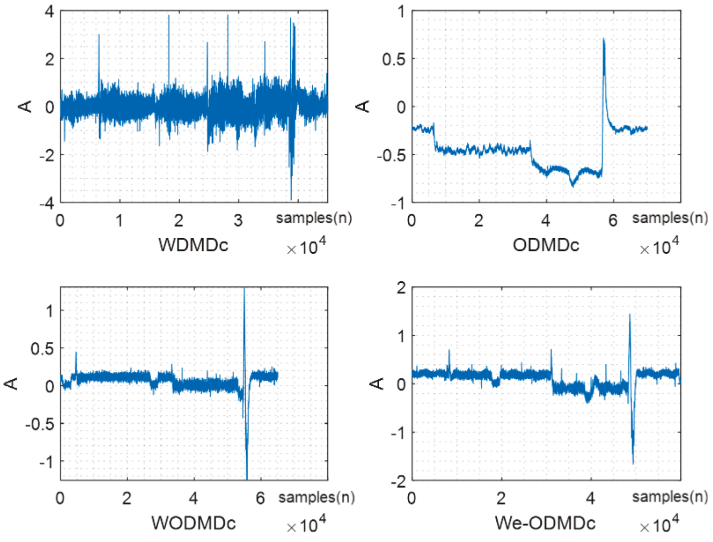

Fig. 19: i_d residuals for the healthy experimental implementation

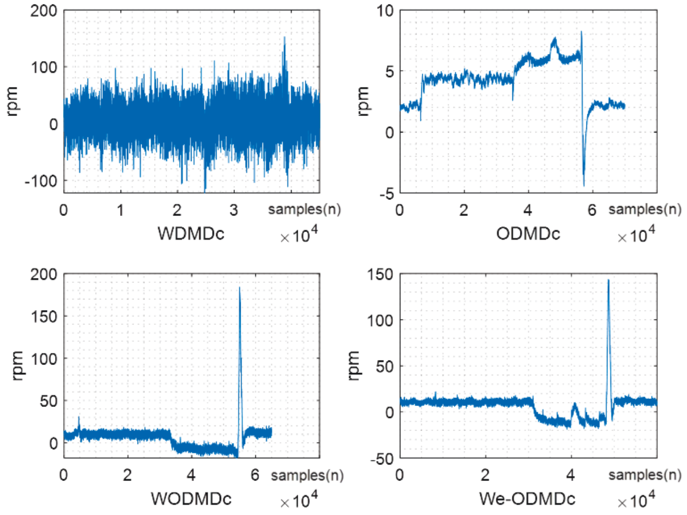

Fig. 20: *wr* residuals for the healthy experimental implementation

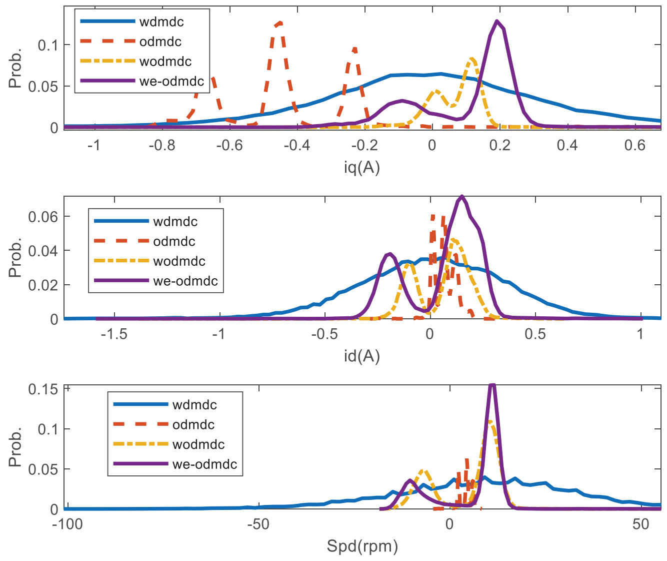

Fig.21: Probability distributions of residuals for healthy experimental data.

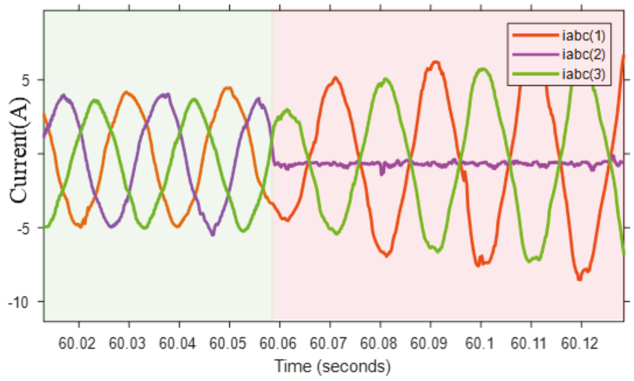

Fig. 22: Phase currents, green shade indicates healthy operation and the red shade indicates faulty operation.

For faulty case a standard test condition is created and applied to all four estimation schemes. The speed is started from 1200 rpm, it is increased to 1500 rpm. Two seconds after speed increment, 2 N.m load torque is applied. While the machine is loaded speed is increased to 1700 rpm and it is

This article has been accepted for publication in IEEE Transactions on Industry Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIA.2022.3231843

reduced to 1200 rpm again. In the next cycle, fault is applied to the system and shortly after experiment is stopped. Following figures depict the overall experiment as well as zoomed in versions of the same quantities. Same sampling conditions as healthy case are used for each DMDc variant and general guidelines given in section IV are followed. Estimation results for the faulty operation is shown in Figures 23 to 26 for the four DMDc variants. It should be noted that by tuning parameters, estimation performances may vary considerably.

An example of the difference in performance is given using We-ODMDc in Fig. 27 where the half-life parameter is tuned. Inherently, there is a trade-off between noise and the sensitivity of the estimation. By adjusting parameters such as the half-life, the estimation can be set more resistant to noise or it can be made more sensitive. Adjusting this variable can change system's resistance to noise and its sensitivity to the sudden changes. For a longer half-life system is more responsive to changes but also open to noise. When half-life is decreased, noise can be suppressed but it becomes insensitive to sudden changes such as faults. For a control application, having a low half-life is beneficial whereas a longer half-life is more useful for condition monitoring.

WDMDc falters in estimating the faulty behavior, the down sampling rate limits the measurable frequency range. It can be thought of as sampling, with 200Hz sampling rate, faster signals cannot be captured. As seen in Fig. 23, WDMDc cannot capture high frequency fault content. Fig. 24 shows WODMDc has higher error in speed estimation.

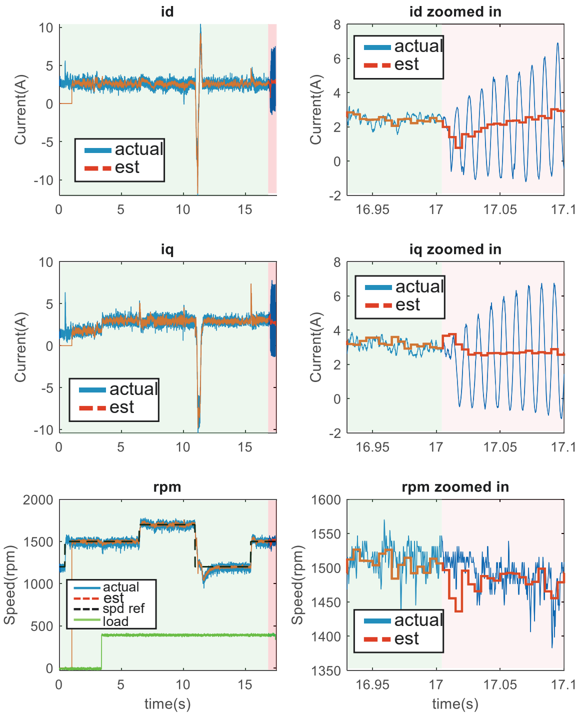

Fig. 23: WDMDc estimates for faulty case, green shade shows healthy state and red shade shows faulty state.

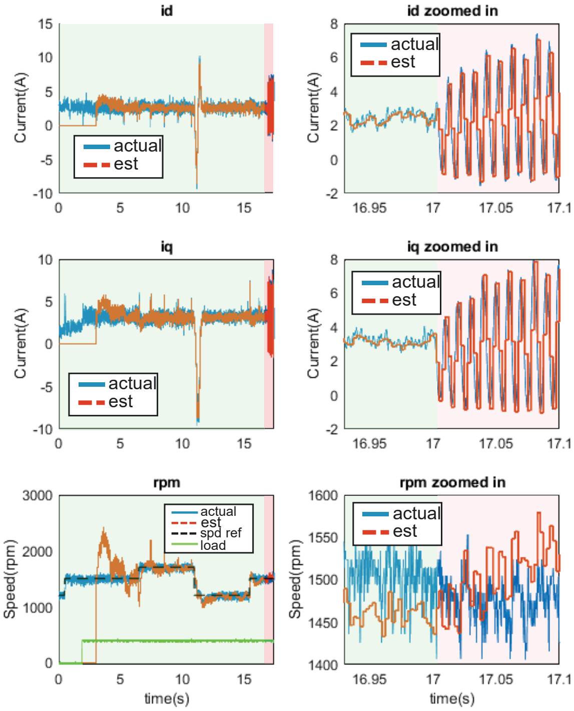

Fig. 24: WODMDc estimates for faulty case, green shade shows healthy state and red shade shows faulty state.

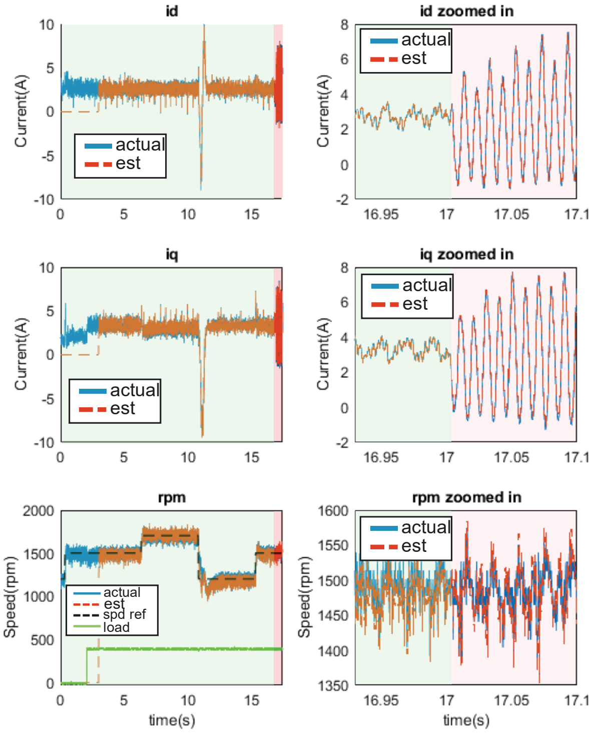

Fig. 25: ODMDc estimates for faulty case, green shade shows healthy state and red shade shows faulty state.

This article has been accepted for publication in IEEE Transactions on Industry Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIA.2022.3231843

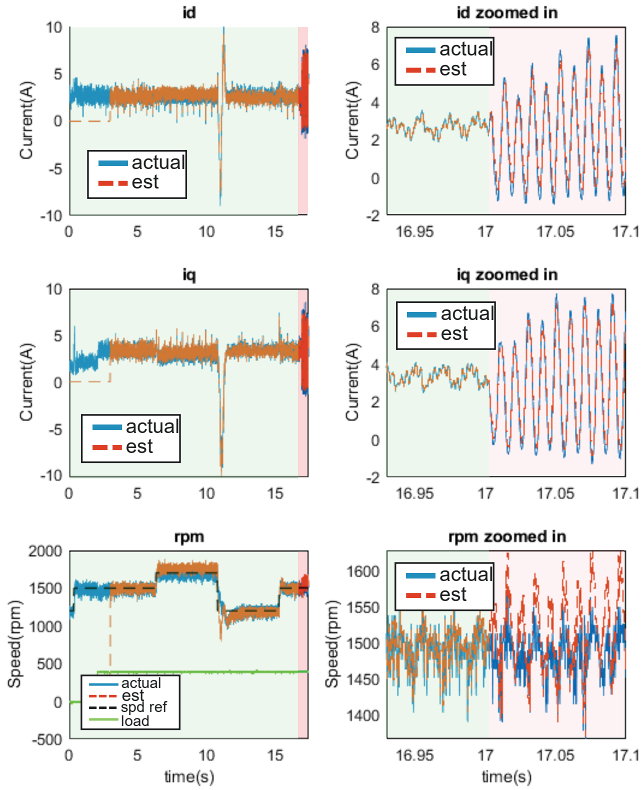

Fig. 26: We-ODMDc estimates for faulty case, green shade shows healthy state and red shade shows faulty state.

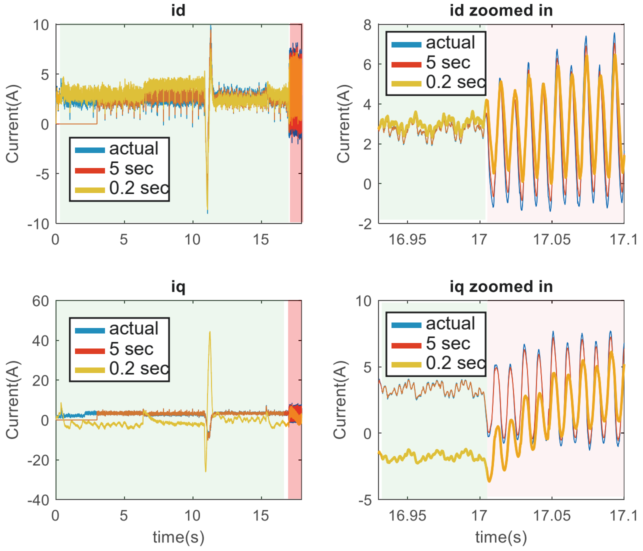

Fig. 27: Parameter tuning example for We-ODMDc. Forget factor is set to 0.2 and 5 seconds respectively.

VI. Discussion, Conclusion and Future Work

In this paper, the strength of the DMDc algorithm in state estimation and prediction in a VSI-fed motor drive is explored. Derivation for the windowed online version is given in detail. Four variations of the algorithm are given. The weighted ODMDc gives a very close estimation while requiring not many resources but falls short if not initiated correctly. The WDMDc does not have any offset problems but it suffers from high computational requirements and high variance. The selection of down sampling rate effects the performance of windowed variants. Online versions show better tracking performance during the fault transient.

The choice of DMDc version can vary between applications. These estimators can be used in other power electronics and motor drive applications as a system identification and estimation tool. The estimation is a powerful method and it is useful for both adaptive control and condition monitoring systems.

The DMDc and its variants are derived and it is tested in exhaustive hardware scenarios. Future work includes utilizing the DMDc for VSI fed IM condition monitoring, including fault detection and diagnosis.

References

[1] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[2] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*,15(1):142–161, 2016.

[3] 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.

[4] 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.

[5] 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,

[6] 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,

[7] Chongke Bi, Ye Yuan, Ronghui Zhang, Yiqing Xiang, Yuehuan Wang, and Jiawan Zhang. A Dynamic Mode Decomposition Based Edge Detection Method for Art Images. *IEEE Photonics Journal*, 9(6), 2017.

[8] E V Filho and P Lopes dos Santos. A Dynamic Mode Decomposition Approach with Hankel Blocks to Forecast Multi-Channel Temporal Series.*IEEE Control Systems Letters*, 3(3):739–744, 2019.

[9] Qiugang Lu and Victor M. Zavala. Image-Based Model Predictive Control via Dynamic Mode Decomposition. *arXiv*, pages 1–20, 2020.

[10] Emilio Barocio, Bikash C. Pal, Nina F. Thornhill, and Arturo Roman Messina. A Dynamic Mode Decomposition Framework for Global Power System Oscillation Analysis. *IEEE Transactions on Power Systems*, 30(6):2902–2912, 2015.

[11] Mohd Zuhaib and Mohd Rihan. Identification of low-frequency oscillation modes using PMU based data-driven dynamic mode decomposition algorithm. *IEEE Access*, 9:49434–49447, 2021.

[12] Deyou Yang, Bo Wang, Guowei Cai, Zhe Chen, Jin Ma, Zhenglong Sun, and Lixin Wang. Data-Driven Estimation of Inertia for Multiarea Interconnected Power Systems Using Dynamic Mode Decomposition. *IEEE Transactions on Industrial Informatics*, 17(4):2686–2695, 2021.

[13] Taku Nonomura, Hisaichi Shibata, and Ryoji Takaki. Extended-kalman-filter-based dynamic mode decomposition for simultaneous system identification and denoising. *PloS one*, 14(2):e0209836, 2019.3

[14] Mojtaba F Fathi, Ahmadreza Baghaie, Ali Bakhshinejad, Raphael H Sacho, and Roshan M D'Souza. Time-resolved denoising using model order reduction, dynamic mode decomposition, and kalman filter and smoother. *Journal of Computational Dynamics*, 7(2):469, 2020.

[15] Maziar S. Hemati, Matthew O. Williams, and Clarence W. Rowley. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids*, 26(11), 2014.

[16] Hao Zhang, Clarence W. Rowley, Eric A. Deem, and Louis N. Cattafesta. Online dynamic mode decomposition for time-varying systems. *SIAM Journal on Applied Dynamical Systems*, 18(3):1586–1609, 2019.

[17] Yadong Yu, Yong Zhang, Sean Qian, Shaofan Wang, Yongli Hu, and Baocai Yin. A Low Rank Dynamic Mode Decomposition Model for

Short-Term Traffic Flow Prediction. IEEE Transactions on Intelligent Transportation Systems, 22 (10): 6547–6560, 2021.3

[18] Hannah Lu and Daniel M Tartakovsky. Prediction accuracy of dynamic mode decomposition. SIAM Journal on Scientific Computing, 42(3):A1639–A1662, 2020.

[19] Kia, S. H., Henao, H., & Capolino, G. A. (2007). “A high-resolution frequency estimation method for three-phase induction machine fault detection.” *IEEE Transactions on Industrial Electronics*, 54(4), 2305–2314.

[20] J. Sun, C. Li, Z. Zheng, K. Wang and Y. Li, "Online Estimation of Per-Phase Stator Resistance Based on DC-Signal Injection for Condition Monitoring in Multiphase Drives," in *IEEE Transactions on Industrial Electronics*, vol. 69, no. 3, pp. 2227-2239, March 2022.

[21] G. Feng, C. Lai and N. C. Kar, "Particle-Filter-Based Magnet Flux Linkage Estimation for PMSM Magnet Condition Monitoring Using Harmonics in Machine Speed," in *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1280-1290, June 2017.

[22] A. Wechsler, B. C. Mecrow, D. J. Atkinson, J. W. Bennett and M. Benarous, "Condition Monitoring of DC-Link Capacitors in Aerospace Drives," in *IEEE Transactions on Industry Applications*, vol. 48, no. 6, pp. 1866-1874, Nov.-Dec. 2012.

[23] Paul C Krause, Oleg Wasynczuk, Scott D Sudhoff, and Steven D Pekarek. *Analysis of electric machinery and drive systems*, volume 75. John Wiley & Sons, 2013.

[24] Zhe Zhang, Muhammed Ali Gultekin, and Ali M. Bazzi. State-space modeling of multi-mode-controlled induction motor drive. In 2021 IEEE *International Electric Machines Drives Conference (IEMDC)*, pages 1–5, 2021.

[25] R.R. Obaid, T.G. Habetler, and R.M. Tallam. Detecting load unbalance and shaft misalignment using stator current in inverter-driven induction motors. In *IEEE International Electric Machines and Drives Conference, 2003. IEMDC'03*., volume 3, pages 1454–1458 vol.3, 2003.

[26] Liu, Yiqi, and Ali M. Bazzi. "A review and comparison of fault detection and diagnosis methods for squirrel-cage induction motors: State of the art." *ISA transactions* 70 (2017): 400-409.

[27] Toliyat, Hamid A., et al. *Electric machines: modeling, condition monitoring, and fault diagnosis*, *p15-19*. CRC press, 2012.

[28] Faiz, Jawad, Mahmud Ghasemi-Bijan, and Bashir Mahdi Ebrahimi. Modeling and diagnosing eccentricity fault using three-dimensional magnetic equivalent circuit model of three-phase squirrel-cage induction motor. *Electric Power Components and Systems*, 43.11 pages 1246-1256, 2015.

[29] A. Sapena-Bano, M. Riera-Guasp, J. Martinez-Roman, M. Pineda-Sanchez, R. Puche-Panadero and J. Perez-Cruz. FEM-Analytical Hybrid Model for Real Time Simulation of IMs Under Static Eccentricity Fault. *2019 IEEE 12th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED),* pp. 108-114, 2019.

[30] Muhammed Ali Gultekin, Zhe Zhang, and Ali Bazzi. Data-driven modeling of inverter-fed induction motor drives using dmdc for faulty conditions. In *2021 IEEE International Electric Machines & Drives Conference (IEMDC)*, pages 1–5. IEEE, 2021. doi: 10.1109/IEMDC47953.2021.9449511.