2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT) | 978-1-7281-9948-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICCASIT50869.2020.9368634

Machine Learning to Optimize Permanent Magnet Synchronous Machines

Zhuoren Ma[a], Ryan Arteaga[b], Muxuan Wang[c], Christine Silveira[d]
Columbia University New York City, USA
[a]zm2336@columbia.edu, [b]ra2910@columbia.edu, [c]mw3347@columbia.edu, [d]cs3876@columbia.edu

***Abstract*—In this paper, we offer a possible approach to estimating motor parameters: Machine Learning. We begin with a simulation of a Permanent Magnet Synchronous Motor (PMSM) model using MATLAB and Simulink. We measure the voltage and current and then send the data to a machine learning model (TensorFlow). From there, we discuss the accuracy of the estimations and future stages of the project. We also use machine learning method to estimate position and speed.**

Keywords—Electric motors, Estimation, Machine Learning, PMSM.

I. Introduction

Satisfying performance of permanent magnet synchronous motor control always requires accurate motor parameters and measurement of position and speed. Usually the parameters include stator resistance, rotor magnetic flux, d-axis and q-axis inductance. These parameters determine the system transfer function, and therefore the controller parameters, including coefficients of PID controllers in current loop, speed loop and so on. However, those parameters vary under different circumstances, so the pre-tweaked coefficients of PID controllers may not work well in experiments. Adaptive control method addresses this problem by measuring motor parameters and updating controller behavior dynamically, but requires accurate motor parameter measurement. For the operation parameters (position and speed), we require an accurate rotor position and speed information in order to deliver high performance control. This can be resolved using a resolver or encoder, but the cost is an important factor to consider. An estimation method could remedy this.

This paper will provide a new method of parameter measurement on the basis of machine learning. Measurement of position and speed with machine learning is also proposed. In Section Ⅱ, factors that bring about parameter estimation will be discussed. Section Ⅲ is literature review. Section Ⅳ describes machine learning method, and Section Ⅴ provides testing results. Section Ⅵ gives discussion about next step. Section Ⅶ is about conclusion.

II. Factors that Influence Motor Parameters

In this section, temperature and magnetic saturation are discussed.

A. Temperature

Temperature of a running motor influences stator resistance linearly,

$$R = R_0\left(1 + \alpha(t - t_0)\right)$$

Where R is resistance at temperature t, and R_0 is resistance at temperature t_0.

In addition, as the temperature increases, permeability of magnetic material will decrease non-linearly, leading to smaller q-axis and d-axis inductance. Flux produced by permanent magnet will also decrease with high temperature. [6].

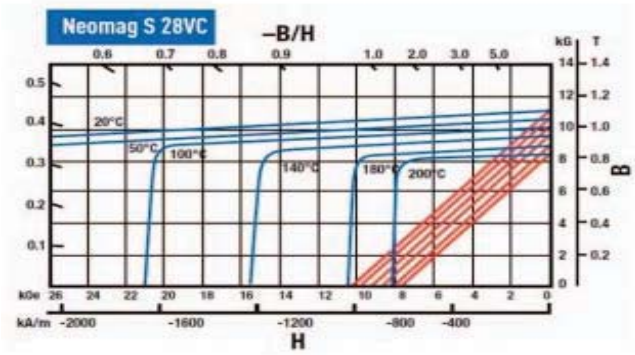

Fig. 1. Temperature influence on magnet flux [7]

B. Magnetic Saturation

Saturation level significantly influences the permeability of the iron core. Driven into high saturation level inductance on d-axis and q-axis will drop. Figure 2 illustrates the function between inductance and dq currents. Due to cross-saturation, inductance on q-axis is also affected by d-axis current, and vice versa.

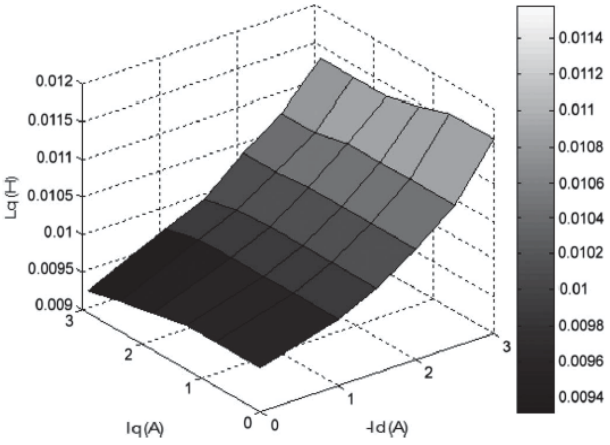

Fig. 2. Saturation influence on q-axis inductance [2]

978-1-7281-9948-1/20/$31.00 ©2020 IEEE

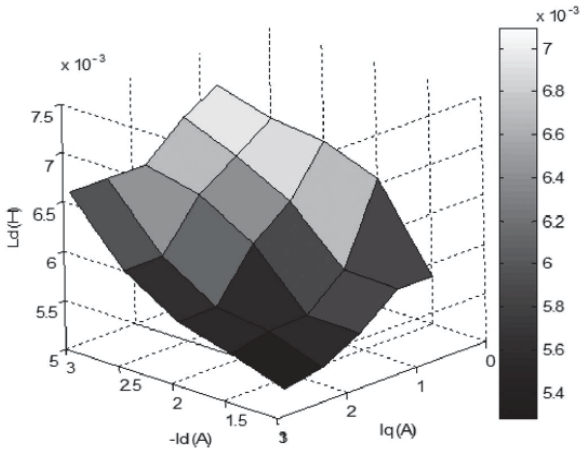

Fig. 3. Saturation influence on q-axis inductance [2]

III. Literature Review

In Professor Preindl's paper, he develops the theory of local identifiability and defines the conditions for position and speed to be identified uniquely based on a sufficiently accurate guess [1]. He sets the task as an optimization problem, based on a least-squares formulation of a dynamic model, with a nonlinear cost function. A dynamic model is derived from the dynamic equations in the dq reference frame of a PMSM, which is shown below.

$$\dot{\lambda}_{dq} = -\omega \mathbf{J} \lambda_{dq} + \bar{v}_{dq} \quad (1)$$

$$\lambda_{dq} = l \circ i_{dq} \approx \mathbf{L} i_{dq} + \psi_r, \quad (2)$$

Note that in equation (1), λdq is the stator flux, vdq is the compensated terminal voltage, J is the 90-degree rotation matrix. Equation (1) shows the relationship between stator flux, speed, and voltage. Equation (2) states the relationship between stator flux, the d and q axis inductances (in diagonal matrix L) and currents and the rotor flux magnitude ψr.

Position dependence is introduced by transforming (1) and (2) into the static αβ reference frame using the orthogonal Park transformation.

$$\bar{v}_{\alpha\beta} = \left(L_\Sigma \mathbf{I} + L_\Delta \bar{\mathbf{P}}(2\theta)\right) \dot{i}_{\alpha\beta} + 2L_\Delta \omega \mathbf{J} \bar{\mathbf{P}}(2\theta) i_{\alpha\beta} + \omega \psi q(\theta), \quad (3)$$

The αβ model can be written in state-space form with our input as vαβ, state as iαβ, and unknowns as position and speed $[\theta, \omega]$ [1].

The cost function can then be checked for convexity. At each sampling point, a numerical solver makes an initial guess and iterates to an optimal guess. The methods the solver uses in its iterations depends on the convexity of the cost function. A strictly convex region can maximize convergence and produce an accurate guess. If the function lacks a strictly convex region or even quasi-convexity, the solver cannot derive a meaning estimate. It will, however, return a guess, which was extrapolated from past samples. This estimation scheme does provide good estimates, but it relies heavily on the convexity of the function.

The current motor parameter estimation strategies can be divided into offline estimation and online estimation.

A. Offline Estimation

Offline measurement usually makes use of data collected under a predefined state. For example, the motor parameters we measured at standstill or steady-state. The results will be stored in a look-up table and the controller uses interpolation to get parameters in certain operation condition [2]. The offline mode does not require intensive calculation during operation, but lacks accuracy because the look-up table only covers limited testing situations.

B. Online Estimation

Common online estimation methods include the Extended Kalman filter, the model reference adaptive system, the recursive least square algorithm and so on [8]. The online method identifies parameters with regard to data collected during operation. This method gives better estimation, but contains complicated matrix calculation, which makes it difficult to synchronize estimator and controller. For example, in [2], an estimation of four parameters is done at 500 Hz, while the sampling or switching frequency of microcontroller can reach 20 kHz. Second, the algorithm has to deal with rank-deficiency problem. [8]

The mathematical model of PMSM can be described from electrical and mechanical equations. Electrical equations are as follows:

$$u_d = R i_d + L_d \frac{di_d}{dt} - w L_q i_q$$

$$u_q = R i_q + L_q \frac{di_q}{dt} + w L_d i_d + w\lambda$$

Where u_d u_q are stator voltages on dq-axes;i_d i_q are stator currents on dq-axes; L_d L_q are dq-inductances. R is stator resistor; $\lambda$is rotor flux. w is electric angular speed.

The mechanical equation is about torque, which is hard to measure, so the mechanical equation will not be discussed here.

There are four parameters to be estimated, but only two equations are available. The algorithm either fixes certain parameters and estimates the remaining, or requires the motor work under a certain state, like maximum torque per ampere (MTPA). The proposed machine learning will feed real-time data into the neural network, aiming to reduce calculations and avoid presumptions about the current online estimation algorithm.

An online parameter estimation algorithm has been proposed by Underwood et al, to estimate stator resistance, rotor magnetic flux, and d- and q-axis inductances [2]. Online estimation algorithms can estimate parameters when new data is present and when the motor is in operation. The estimated values can vary with time. Offline estimation methods require us to gather all input and output data first and then provide the estimates, which are time-invariant. Underwood's paper first discusses the PMSM Modeling and parameter sensitivity, such as temperature effect on stator resistance, magnetic flux, and how events such as magnetic saturation also affects the performance of the motor.

Underwood's proposed algorithm is formed by combining both a fast and slow recursive least square (RLS) algorithm. The algorithm gives equal weight to all measurements, just as the offline least square algorithm does, with an assumption

that the estimated parameters are constant for the whole duration of the experiment.

Underwood discussed his simulation and experimental results. The fast RLS algorithm begins with an initial estimate. Once it converges, meaning the system is in steady-state, the slow RLS algorithm uses the estimates from the fast RLS algorithm as a starting point and then converges to an optimal estimate. There was an error of approximately 30% on the initial estimate of stator resistance, which does have an effect q-axis inductance estimate. However, the estimates do converge, and the estimates improve. The results do validate the algorithm's effectiveness. While the paper's focus was on interior permanent magnet (PM) machines, the author does explain that the concept and algorithm can be applied to other PM machines and types.

IV. Methodology

A. Our approach

We started our research into PMSM with a comprehensive literature review. From the literature, we learned what past papers have used as optimal parameters and why certain parameters must fall into a particular range. In particular, PM flux and Inductance have constraints. PM flux varies with temperature and high temperatures can cause demagnetization. Inductance can saturate at high current levels.

With this knowledge, we created training data that would take our constraints into account. We varied conditions within our model to create a wide range of training data that changes speed and torque. Using these profiles, we predict what the ideal parameters would be under similar circumstances. Corresponding to practical situations, all of the parameters are between -20% and 20% of nominal values. Similarly, when finding the ideal speed and position, the constraints were accounted for.

The model in Simulink was created by previous members of Professor Priendl's lab, but many changes have been made to it. The speed and torque have both been stepped to create different profiles as a variety of training data for the machine learning code. In each profile, the speed is brought high until it stabilizes, is then brought negative and increased again. An example of one such profile can be seen in Figure 4. Matlab code has been written to take multiple data profiles at once.

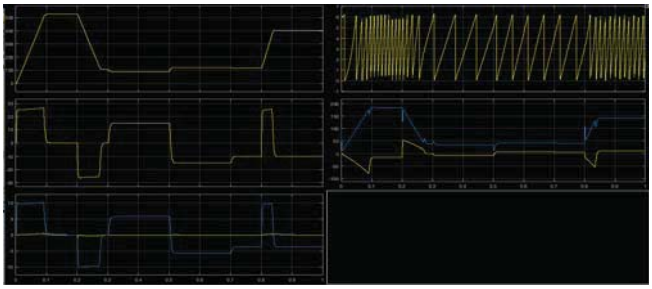

Fig. 4. Above omega, torque, current, theta, and voltage are graphed in order from top to bottom.

Once the data is collected, it is used to train a machine-learning algorithm. We divided the data equally into two sets. One set of the data is used for training, and the other set is used for testing and comparing. Each set is randomly selected. Below is the algorithm to train our model. Library Keras is used for estimation.

```
def build_model():
  model = tf.keras.Sequential([
    layers.Dense(32, activation='relu', input_shape=[len(train_dataset.keys())]),
    layers.Dense(32, activation='relu'),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
  return model


# Display training progress by printing a single dot for each completed epoch
class PrintDot(keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs):
    if epoch % 100 == 0: print('')
    print('.', end='')

EPOCHS = 1000

model = build_model()

# The patience parameter is the amount of epochs to check for improvement
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)

history = model.fit(train_dataset, train_label, epochs=EPOCHS,
                    validation_split = 0.2, verbose=0, callbacks=[PrintDot()])
```

Fig. 5. Machine Learning Code Snippet

We kept increasing the number of data points and epochs until the error lowered. In the parameter estimation case, every parameter is varied with four values. In total, there are 256 profiles. We use 102,400 data points to estimate four parameters. Among them, 80% are used for training; 20% are for testing. u_d u_q i_d i_q w $i'_q i'_d$ are inputs of the neural network and $i'_q i'_d$ are currents from the last sample. The network is trained with 60,000 epochs. In the Speed & Position case, we used 1,500 data points and 1,000 for epochs for each testing condition. We tested 6 test conditions. We obtained a mean absolute error and mean square error of approximately 1 percent. In Speed & Position, I_a, I_b, I_b_delay, I_a_delay, Va and Vb are used as inputs, while Theta and Omega are output estimates.

B. Division of Labor

Muxuan and Christine are focusing their efforts on the parameter optimization. Christine has been focused on collecting data from the Simulink model. She has written code to take multiple data profiles at once. We currently have 256 profiles that vary the parameters within the ranges discussed in the literature. Muxuan has been working on editing and successfully running the machine learning code. She has seen improvement in the operation and results of the algorithm.

Ryan and Zhuoren are focusing their efforts on the speed and position optimization. Ryan has been focused on discovering the relationship between parameters and Speed & position. He has been able to create 6 profiles that vary torque and speed, while avoiding bad operation points (i.e. zero speed). He is currently developing code that can vary step functions used to change the motor parameters. Zhuoren is working on modifying the machine learning code to be used for estimating motor parameters and is able to obtain accurate predictions with the algorithm

V. Results

A. Optimization of Speed and Position

We demonstrate one simulation trial. The trial consists of us initially stepping the motor's speed to 13.33 radians per second (rad/s), then 6.67 rad/s at 200ms, and then 0 at 900ms. We also set the load torque to 30Nm at 400ms, 15Nm at 600ms, and then 0 at 900ms. Figure 6 shows the output waveforms. Note that the current and voltage waveforms are measured with respect to the αβ frame.

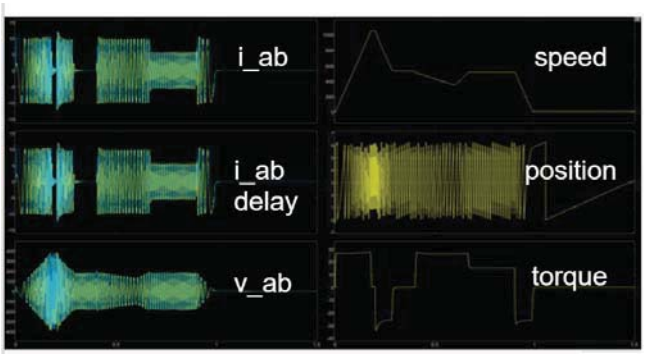

Fig. 6. Output waveforms for current and voltage, speed, position, torque for Trial 1

We then extract the data as a csv file and insert it into the machine learning algorithm. The total set of data points was 1,500, where 750 of them were randomly selected to be the training set and the rest were in the testing set. 1,000 epochs were used when running the algorithm. The network was 32*32. Figures 7 show the absolute and square mean error for theta, which is position, respectively. Figures 9 and 10 show the models prediction and error prediction for position, respectively. Loss is the mean-squared error. While torque data was collected, it was not needed for the machine learning model.

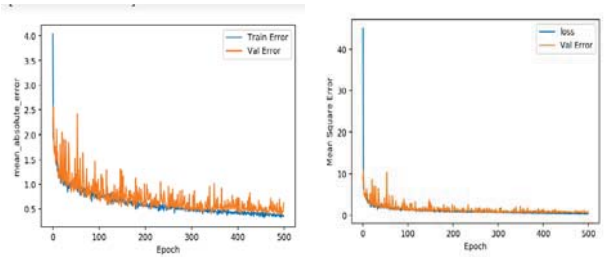

Fig. 7. The absolute and square mean error for position.

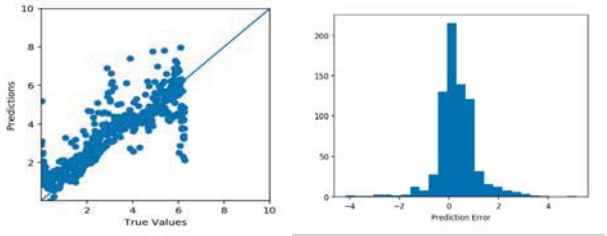

Fig. 8. Prediction and Prediction Error for Position

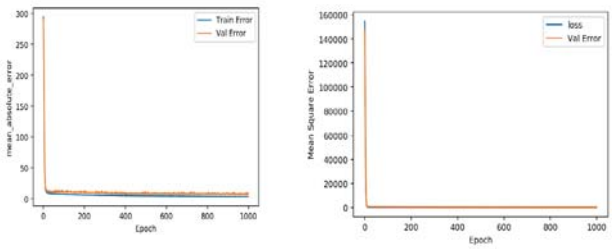

Fig. 9. The absolute and square mean error for position

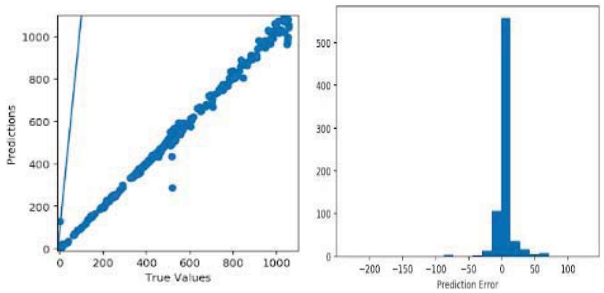

Fig. 10. Prediction and Prediction Error for Position

There is some error between predicted and actual position values, which are described in Table I. The reason for the error is due to not having enough data to train the machine learning and not using more epochs. The position function is discontinuous, especially when it shifts from 6.28 to 0 radians and vice versa. One way to remedy this is to estimate a continuous trigonometric function (e.g. sine) and derive the position with the inverse function. The Square mean square error is high for both Position and Speed. The reason for the high mean square error is the high errors in the first few trails. Take Speed as an example, the error for the first set of data is 140,000. Since we only have 1,000 epochs, this will increase our mean square error by 140. If we have more epochs, the square mean error will be decreased.

TABLE I. MEASUREMENT ERRORS FOR POSITION AND SPEED

	Position	Speed
Mean Absolute Error	0.3883	6.8693
Mean Square Error	3.9017	261.8

B. Optimization of Parameters

In our first iteration of the experiment, we try to estimate one parameter, L_q, with a nominal value 12.9 mH.

Settings for training are shown in Table II.

TABLE II. SETTINGS FOR MODEL TRAINING

Network	64*64
Epoch	60000
Data points	1980
Training percent	80%
Loss	Mean-squared-error

The final predictions results and histogram of errors are shown below. The mean-absolute-error of prediction is 0.001, corresponding to about 7.7% error. The mean-squared-error is 1.6505e-6.

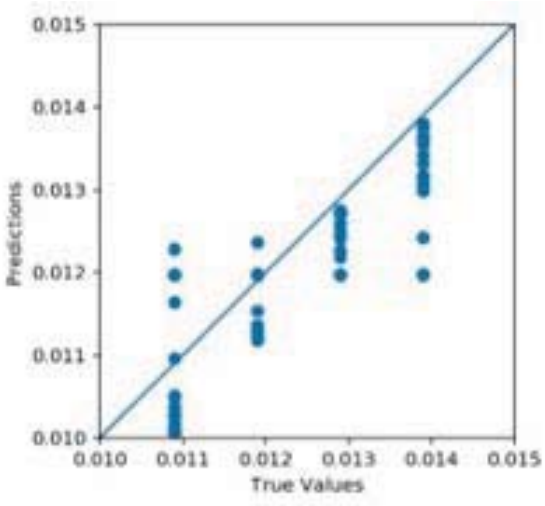

Fig. 11. Predictions and true values of Lq

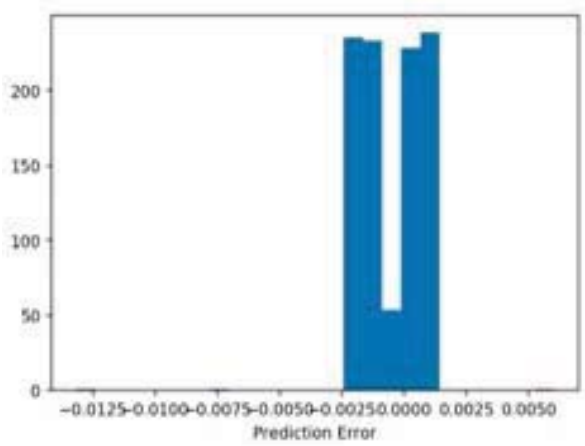

Fig. 12. Histogram of prediction errors

With the same settings, two parameter estimations are implemented. Stator resistance R (0.4Ω±20%), and λ (0.3491 wb±20%) are used as outputs. The total datasets are 256,00.

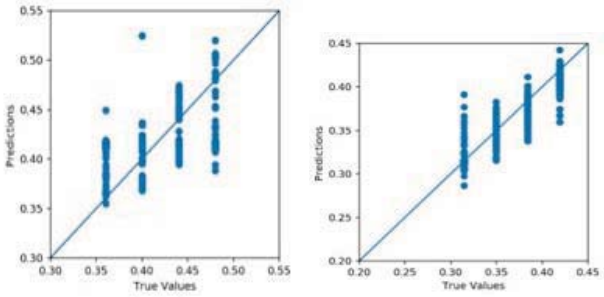

Fig. 13. Predictions and errors of resistance and flux

The mean-absolute-error and mean-squared error are as follows.

TABLE III. RESULTS OF PARAMETER ESTIMATION

	mean-absolute-error	mean-squared-error
R	0.0386 (9.6%)	0.00838 (2%)
λ	0.002 (0.5%)	0.00014 (0.04%)

Finally, four parameter estimation is tried. The total data points are 102400.

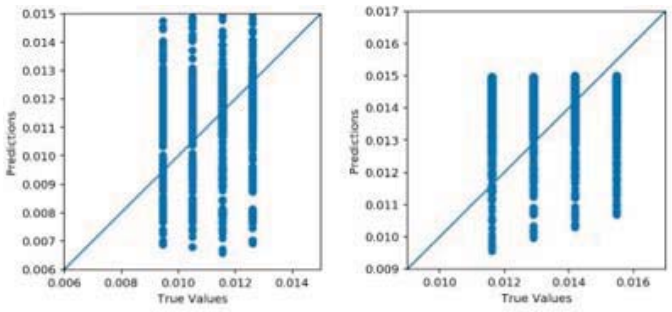

Fig. 14. Predictions of dq-axis inductance

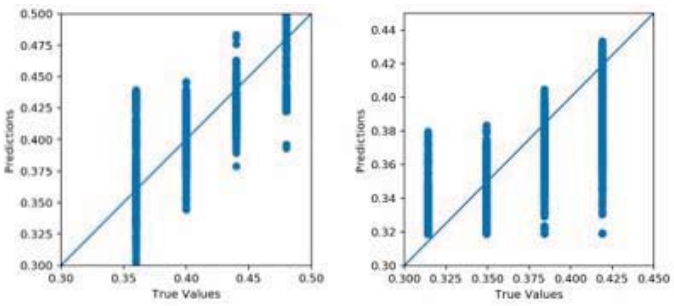

Fig. 15. Predictions of resistance and flux

TABLE IV. RESULTS OF FOUR PARAMETER ESTIMATION

	mean-absolute-error	mean-squared-error
R	0.0389 (9.7%)	0.00197 (0.5%)
λ	0.0077 (2%)	0.000144 (0.04%)
L_q	0.0014 (10.8%)	2.885e-6 (0.0224%)
L_d	0.00127 (12%)	2.38e-6 (0.0227%)

VI. NEXT STEPS

A. Model

The model will continue to be used for collecting training data. Work can be done to streamline the code and improve run time, but unless a user wishes to drastically change the types profiles being made, it is complete for our purposes.

In future applications, someone may choose to focus on the data of a specific behavior within the profile. The code can be rewritten to save data exclusive between certain time marks such as, between 300 and 500 ms for which the model runs.

B. Optimization of Parameters

We can see from the parameter estimation results that there exists a large error, especially when we try to estimate four parameters.

In the future, our goal is to reduce the error as much as possible and to continue to train the model on varying data profiles. In the future, we should first pre-process data, such as normalization, because different data, whether input or output, have drastically different variation range. For example, d-axis inductance estimation has poor performance in the last training, most likely because id is close to zero under the control algorithm. We can also try to convert dq current from the last sample into current derivation. Second, all the above-mentioned training results for parameter estimation are under 60,000 epochs. According to trial and error, more iterations

will reduce the error, so testing with more epochs is another option. We are confident that with more time and data our algorithm would be able to predict the optimal parameters for a wide range of speed and torque variations. Some of our error is also the result of varying 4 parameters at once. It will be valuable in future iterations to continue studying the effect that each of these parameters has on each other.

C. Optimization of Speed and Position

There are more steps to be taken with regards to our prediction errors.. First, we must continue to gather more data. This includes using more steps during simulation, randomize the behavior of those steps, and run the model for more than one minute. Another aspect to consider is the discontinuity of position. Position varies from 0 to 6.28 radians and it will step down to zero once it reaches 6.28 radians. The discontinuity contributes to the error in our position predictions. One method is to use trigonometric functions (e.g. sine) that are continuous. If we can predict sin (theta), we can then determine theta using the arcsine function. We also kept the machine parameters constant, which can vary. Once we start to change one or more parameters, we may likely introduce more error into our position and speed predictions.

VII. Conclusion

In order to determine a valid estimation method for PMSM parameters, we have conducted two simulation trials. One trial is designed to estimate the motor parameters given position and speed information. Another trail to estimate the operation parameters if we know the motor parameters. The results of both trials show that we are able to predict the unknown parameters, but more work is needed to reduce error and to begin testing on real, physical motor.

Acknowledgment

We would like to thank Professor Mattias Preindl for allowing us to work in his lab and providing weekly guidance. We would also like to thank Michael Eull for providing insight into how to use the model and code and Youssef Fahmy for his previous work on the machine learning code

References

[1] M. Preindl, Convergence Properties of Direct Position and Speed Estimation in Synchronous Motor Drives. IEEE International Symposium on Sensorless Control for Electrical Drives (SLED). 2019.

[2] S. Underwood, I Husain. Online Parameter Estimation and Adaptive Control of Permanent-Magnet Synchronous Machines. IEEE Transactions on Industrial Electronics, vol. 57, no. 7, pp. 2435-2443. July 2010.

[3] W. Wang, N. Brady, C. Liao, Y. Fahmy, E. Chemali, A. West, M. Preindl. High-Fidelity State-of-Charge Estimation of Li-Ion Batteries using Machine Learning. arXiv: 1909. 02448v1. 2019.

[4] P. Pillay, R. Krishnan. Modeling, Simulation, and Analysis of Permanent-Magnet Motor Drives, Part I: The Permanent-Magnet Synchronous Motor Drive. IEEE Transactions on Industry Applications, Vol. 25, No. 2. March/April 1989

[5] O. Bilgin and F. A. Kazan, "The effect of magnet temperature on speed, current and torque in PMSMs," 2016 XXII International Conference on Electrical Machines (ICEM), Lausanne, 2016, pp. 2080-2085.

[6] V. Bobek, “PMSM Electrical Parameters Measurement,” 2013, Freescale Semiconductor, Inc.

[7] Bilgin, Osman & Kazan, Fatih. The effect of magnet temperature on speed, current and torque in PMSMs. 2080-2085. 10.1109/ ICELMACH.2016.7732809. 2013

[8] K. Liu, Q. Zhang, J. Chen, Z. Q. Zhu and J. Zhang, "Online Multiparameter Estimation of Nonsalient-Pole PM Synchronous Machines With Temperature Variation Tracking," in IEEE Transactions on Industrial Electronics, vol. 58, no. 5, pp. 1776-1788, May 2011