USING PRINCIPAL COMPONENT ANALYSIS AND BINARY CLASSIFICATION TO DETECT STATIC ECCENTRICITY FAULTS IN INDUCTION MOTORS

Nicolas A. Krause[1], Jake J. Malone[1], Meg J. B. McPherson[1], Frederik K. van Wingerden[1], Martin J. Z. Yong[1], Ilamparithi Thirumarai-Chelvan[1]

[1]*Department of Electrical Engineering* University *of Victoria, Victoria, Canada*

nkrause@uvic.ca, jakemalone@uvic.ca, margaretmcpherson@uvic.ca, frerik@uvic.ca, myong@uvic.ca, ilampari@uvic.ca

Keywords: ECCENTRICITY, FAULT DIAGNOSIS, INDUCTION MOTOR, MACHINE LEARNING

Abstract

Static eccentric faults can cause serious damage to induction motors if left undetected. Standard methods identify static eccentric faults by performing frequency or harmonic analysis of the motor currents and by monitoring for the principal slot harmonics. However, the principal slot harmonics arise only if the rotor bars and the pole-pair number of the motor conform to certain relationship. To overcome this limitation, this paper describes two machine learning methods to detect static eccentric faults using either voltage and/or current data. Principal Component Analysis, and a Binary Linear Classifier are used independently, to classify experimental data from both a healthy, and faulty motor. Both machine learning algorithms were written in MATLAB and their performance was evaluated using a standard confusion matrix.

1. Introduction

In recent years electric cars have exploded in popularity. Manufacturers such as Tesla, VW and Ford are producing mass market electric cars [1]. A key component of any electric car is the motor that propels it. Like any device, electric motors can wear out and break. A key class of fault for electric motors is called an eccentricity fault. Broadly, this is when the stator and/or rotor of the motor are rotating off a common central axis [2]. If the motor continues to operate in this condition serious damage can result. There are three types of eccentricity faults, but this paper focuses on a single type, the static eccentricity fault. This is when only the rotor is off axis, but the stator is not [2].

Standard static fault detection methods in induction motors tend to rely on harmonic or frequency analysis of the motor currents [3]. Such schemes suffer from the limitation that the principal slot harmonics (PSH) arising due to static eccentricity faults are dependent upon certain pole-pair and rotor bar number combination [3]. To overcome this limitation, fault detection using machine learning (ML) techniques is explored in this paper. Machine learning has exploded in the past decade as a research subject [4]. With the advent of inexpensive sensors to record data, and more powerful computational techniques, machine learning techniques have been broadly adopted across numerous industries [4]. Some aspects of the car industry have made powerful use of machine learning methods in innovative ways; the current work by numerous companies to build self-driving cars is one such example. In particular, they often use advanced techniques such as neural networks to allow a car to see and predict [4]. However, machine learning techniques have not been applied in the same manner to the problem of static eccentric fault detection in induction motors.

Machine learning based fault detection methods hold the promise of being more robust, cheap, and easy to implement. Two primary machine learning techniques are implemented and examined in this paper. Principal Component Analysis (PCA) and binary linear classification are used to determine if experimental current and voltage data from an induction motor indicate that the motor has a static eccentric fault. Both methods are well understood mathematical techniques and are easy to implement using standard programming tools such as MATLAB. Instead of requiring large expensive computational setups, both methods can be implemented on single widely available commercial processors.

2. Methodology

The main design objective of this research is to use PCA and binary classification to successfully detect static eccentricity fault in an induction motor. For this research, the algorithms use experimental data from an induction motor that was made to operate with and without static eccentricity fault. Since the ground truth value of the sample data is known, supervised classification techniques can be used to create the algorithm [5]. A brief description of the two approaches used in this research is given below.

2.1. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is an algorithm that focuses on reducing the dimensionality of datasets by transforming large sets of data into smaller ones while preserving as much of the original data as possible [6]. This is achieved by creating new uncorrelated variables called principal components. Principal components are a collection of points and unit vectors based on a best-fit line that minimizes the average squared distance from each point to the line. Using these principal components, the dataset is simplified into an eigenvalue/eigenvector problem within the data's covariance matrix [6].

Principal components are constructed by the algorithm in such a way that the first principal component accounts for the largest possible variance in the data set [6]. The linear combination with the maximum variance is determined by calculating the average of the squared distances from the projected points onto the established origin.

The data used in the PCA algorithm is first modified using a standard transform. The alpha-beta transform (1) creates two synthetic currents (I_α, I_β) or voltages (V_α, V_β) from three-phase current (I_A, I_B, I_C) or voltage (V_{AB}, V_{BC}, V_{CA}) data [7].

$$\begin{bmatrix} I_\alpha \\ I_\beta \\ I_\lambda \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} I_A \\ I_B \\ I_C \end{bmatrix} \quad (1)$$

This simplifies the analysis of data, and is a technique that has been in standard use by electrical engineers for many years [5]. As a result, PCA expects to find two principal components. After the alpha-beta transform, PCA algorithm can be broken down into three steps [8]:

1. Standardize the data set by subtracting the mean $(\bar{\alpha}, \bar{\beta})$ of the data dimensions from the alpha-beta transform (2).

$$[\boldsymbol{\alpha} - \bar{\alpha} \quad \boldsymbol{\beta} - \bar{\beta}] \quad (2)$$

2. Determine correlation of data by computing covariance matrix. For a two-dimensional data the covariance matrix is shown below (3).

$$\begin{bmatrix} cov(\alpha, x) & cov(\alpha, y) \\ cov(\beta, x) & cov(\beta, y) \end{bmatrix} \quad (3)$$

3. Identify principal components by computing eigenvectors and eigenvalues. The eigenvectors (v_1, v_2) correspond to the two principal components and the eigenvalues (λ_1, λ_2) correspond to the variance value for each principal component (4), (5).

$$\boldsymbol{v_1} = \begin{bmatrix} v_{x1} \\ v_{y1} \end{bmatrix} = eigenvector\ 1,\ \lambda_1 = eigenvalue\ 1 \quad (4)$$

$$\boldsymbol{v_2} = \begin{bmatrix} v_{x2} \\ v_{y2} \end{bmatrix} = eigenvector\ 2,\ \lambda_2 = eigenvalue\ 2 \quad (5)$$

MATLAB's PCA function is used to analyse the transformed training datasets. This generates two healthy basis vectors and two faulty basis vectors. In the alpha-beta transform plot, the faulty dataset is expected to have a rotational offset from the healthy dataset. This feature is present in the basis vectors as well. The classification vector is then generated by adding a faulty vector to a healthy vector. As per the laws of vector addition this creates a vector halfway between the two. To classify a test data sample, PCA analysis is performed on the sample, resulting in two basis vectors which are compared to their respective threshold vectors. This comparison is performed by looking at the rotation angle theta, of the midpoint vector and the new sample vector. The classification rules depend on the location of the initial faulty and healthy vectors and the midpoint vector they generate.

An automated classifier is then written to identify healthy and faulty samples using the method described above. The program first performs an alpha-beta transformation on the healthy dataset and faulty dataset. Next, the principal components are determined and a midpoint between the two is found to establish a threshold. Classification rules are determined via manual examination of the threshold vectors generated from the training data. Finally, the unknown dataset is fed into the algorithm, where the PCA function compares the test data linear combination with the threshold and classifies accordingly.

2.2. Binary Classification - Logistic Softmax Cost Function

For the research problem under consideration, since there are only two possible outcomes, namely healthy and faulty, binary classification can be used to detect them [4]. A binary classification equation can classify input samples ($\boldsymbol{x}$), as either Class A (Healthy) or Class B (Faulty) [4]. To do this, an equation multiplies the input samples ($\boldsymbol{x}$) by some weights ($\boldsymbol{w}^*$) and adds a bias ($\boldsymbol{b}^*$) [4]. If the answer is positive, (1) will classify the output ($\tilde{\boldsymbol{y}}$), to be Class A (Healthy) [4]. If the answer is negative, (6) classifies the output ($\tilde{\boldsymbol{y}}$) as Class B (Faulty) [4].

$$\tilde{y} = sign(w^{*T}x + b^*) \quad (6)$$

For this research the input samples being used are the motor's line-to-line voltages and the line side currents. To determine if the fault was detected by monitoring the voltage or current two binary classification equations were used: one for voltages and one for currents.

To create the weights ($\boldsymbol{w}^*$), and bias ($\boldsymbol{b}^*$) for the binary classification equations, the machine learning algorithm needs

to find a local minimum ($\hat{\boldsymbol{w}}^*$) of the Logistic Softmax Cost Function ($\boldsymbol{E_L}(\hat{\boldsymbol{w}})$) using the known sample data (7) [4].

$$E_L(\hat{w}) = \frac{1}{P}\sum_{P=1}^{P} \log(1 + e^{-y_P \hat{w}^T \hat{x}_P}) \quad (7)$$

where

$P = The\ number\ of\ samples$

$y_P = Known\ classification\ for\ each\ training\ sample$

$x_P = Input\ samples\ for\ training/testing$

$\hat{x}_P = \begin{bmatrix} x_p \\ 1 \end{bmatrix}$

$\hat{w}^* = Local\ Minimum = \begin{bmatrix} w^* \\ b^* \end{bmatrix} = \begin{bmatrix} Weights \\ Bias \end{bmatrix}$

To use (7), the raw sample data needs to be in the correct format. To do this, the data is broken down from samples of 54000 points/phase into smaller training/testing samples. For this project, the raw data was broken up into sample groups of 240 points/phase.

Figure 1 shows how the raw input voltages of a three-phase system; however, to ensure that all the samples are being compared correctly, the samples need to be aligned and normalized. To align the samples, the system finds the maximum value of a sample. It then shifts all the sample data until the maximum value aligns with time origin. Figure 2 shows the voltage data after it has been aligned and normalized.

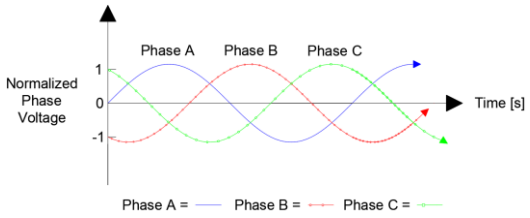

Figure 1: Raw Sample Data – Normalized Phase Voltages

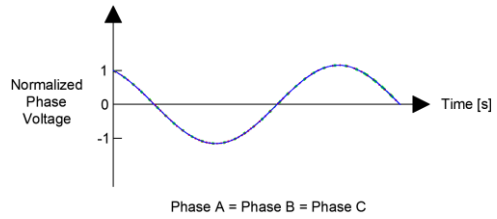

Figure 2: Normalized Phase Voltages – Aligned to Zero

Once the data has been manipulated into the correct format, the machine learning algorithm will use the samples to create a set of weights and bias [4]. A single input sample for the machine learning algorithm needs a column vector made up of input parameters (x) shown in (8).

$$\boldsymbol{x} = [\boldsymbol{x_0} \quad \boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \ldots \quad \boldsymbol{x_n}]^T \quad (8)$$

Figure 3 displays how the voltage is broken down into input data points. Each point along the curve acts as an input parameter.

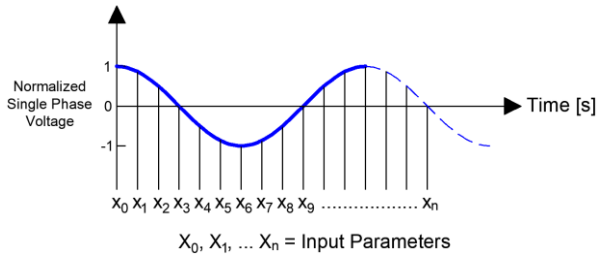

Figure 3: Visual Representation of Input Parameter of a Single-Phase Voltage Sample

After a single sample is created, multiple samples can be combined creating a matrix of samples [4]. Equation (9) shows how single samples ($x_0\ to\ x_M$) is combined to create a matrix of multiple samples (x_P).

$$\boldsymbol{x_P} = [x_0 \quad \cdots \quad x_M] = \begin{bmatrix} x_{00} & \cdots & x_{M0} \\ \vdots & \ddots & \vdots \\ x_{0n} & \cdots & x_{Mn} \end{bmatrix} \quad (9)$$

where

$\boldsymbol{x_P} = Multiple\ samples\ for\ training/testing$

$n = The\ number\ of\ parameters$

$M = The\ number\ of\ samples$

Along with the sample data ($\boldsymbol{x_P}$), the system requires a vector containing the known classification values ($\boldsymbol{y_P}$) of the sample data ($\boldsymbol{x_P}$). For this work a sample is considered healthy if it is labelled with a one (1). If it is faulty, it is labelled with a negative one (-1). It is important that each known classification ($\boldsymbol{y_P}$) in (10) is aligned with its corresponding sample data.

$$\boldsymbol{y_P} = [y_0 \quad \ldots \quad y_n] \quad (10)$$

$$y_0\ to\ y_n = \begin{cases} 1; & if\ the\ sample\ is\ healthy \\ -1; & if\ the\ sample\ is\ faulty \end{cases}$$

Once the sample data has been prepared it can then be applied to (7), the Logistic Softmax Cost Function ($\boldsymbol{E_L}(\hat{\boldsymbol{w}})$). The machine learning algorithm will use the Newton algorithm method along with backtracking line search to find a local minimum of the Logistic Softmax Cost Function ($\boldsymbol{E_L}(\hat{\boldsymbol{w}})$). This local minimum ($\hat{\boldsymbol{w}}^*$) equals the weights ($\boldsymbol{w}^*$), and bias ($\boldsymbol{b}^*$) for the binary classification in (11).

$$\hat{w}^* = Local\ Minimum = \begin{bmatrix} w^* \\ b^* \end{bmatrix} \quad (11)$$

Now that the weights ($\boldsymbol{w}^*$), and bias ($\boldsymbol{b}^*$), have been generated, the system can now perform binary classifications on new data using the binary classification (12) below.

$$\tilde{y} = sign(w^{*T} x + b^*) \quad (12)$$

3. Results & Validation

The performance of the two algorithms can be measured by using the following two methods:

- Confusion matrix
- Performance check using test data sets

3.1. Confusion Matrix

A confusion matrix is a visualization tool used to evaluate the performance of a machine learning classification algorithm. It consists of a symmetrical matrix that shows the number of correct and incorrect predictions that have been made [4]. The formula representing a confusion matrix is shown below (13) [4].

$$Confusion\ matrix = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad (13)$$

Where A, B, C and D in the confusion matrix are defined by:

A = number of samples from class 1 (healthy) that are classified correctly;
B = number of samples from class -1 (faulty) that are incorrectly classified to class 1 (healthy);
C = number of samples from class 1 (healthy) that are incorrectly classified to class -1 (faulty);
D = number of samples from class -1 (faulty) that are classified correctly.

The percent accuracy can be calculated from the confusion matrix using (14) [4]:

$$Classification\ accuracy = \frac{A+D}{A+B+C+D} \times 100 \quad (14)$$

The percent accuracy indicates how accurate the algorithm is at classifying. A higher percent accuracy indicates an algorithm that classifies accurately [4]. It should be noted that for this research a high or perfect accuracy of 100% is expected as the data sets include only a single class of fault. In a real-world scenario, multiple faults or errors could occur simultaneously, which would interfere with properly classifying the samples.

3.2. Performance Check Using Various Test Data Sets

Five different test data sets obtained from a 45-rotor bar, 4 pole, 3-phase induction motor operating under different faults and load conditions were provided to test the PCA and binary classification. Both the training and testing data sets were arranged in a m by n matrix (15). The first three columns are line-to-line motor voltages (V_{AB}, V_{BC}, V_{CA}) and the last three columns are motor stator currents (I_A, I_B, I_C).

$$\begin{bmatrix} V_{AB_{11}} & V_{BC_{12}} & V_{CA_{13}} & I_{A_{14}} & I_{B_{15}} & I_{C_{16}} \\ V_{AB_{21}} & V_{BC_{22}} & V_{CA_{23}} & I_{A_{24}} & I_{B_{25}} & I_{C_{26}} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ V_{AB_{m1}} & V_{BC_{m2}} & V_{CA_{m2}} & I_{A_{m4}} & I_{A_{m5}} & I_{C_{mn}} \end{bmatrix}_{m \times n} \quad (15)$$

The test data sets are distinct from the training data sets used for development of the classification algorithm. The classification, or ground truth, of each sample in the test and training data set is already known. Each data set consisted of different percentage of static eccentricity fault (0%, 20% and 40%) and motor loading (0%, 50% and 100%).

3.3. PCA Results

When running test files on the PCA algorithm it was able to obtain an 100% testing accuracy. Figure 4 shows the healthy versus faulty current with PCA basis vectors obtained from running MATLAB's PCA function. The rotation present in the faulty dataset is clearly represented by the offset of the faulty basis vector from the healthy basis vector. Also shown are the two basis vectors of each healthy and faulty analysis.

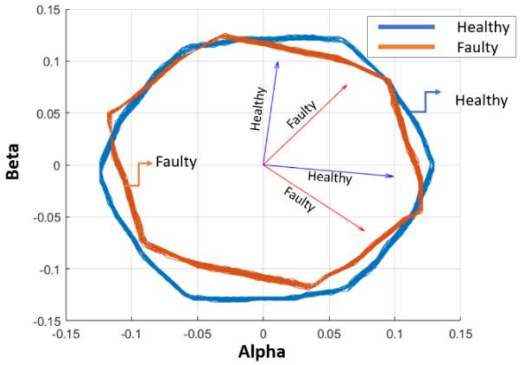

Figure 4: Healthy vs. faulty current with PCA basis vectors.

Figure 5 shows the PCA voltage threshold vector obtained from the classifier written to identify the healthy and faulty samples. The threshold vector (VT) shown is used to compare the two-basis vectors that were obtained. From this the classification rules are determined, and the classification is performed.

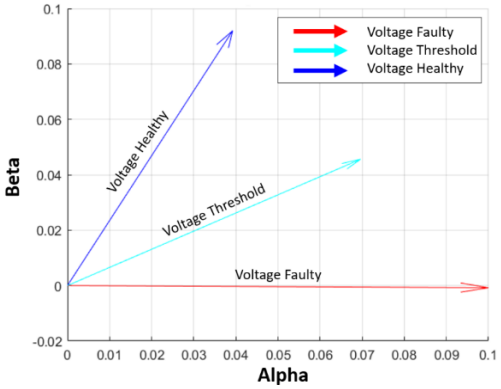

Figure 5: PCA voltage threshold vector

3.4. Binary Classification Results

Table 1 shows the results obtained from using binary classification on the two training datasets: healthy and faulty. It is shown that all voltage and current datasets are classified correctly at no load conditions.

Table 1: Training dataset results

Test	Fault	Loading	Testing Accuracy
1	0%	0%	100% voltage & 100% current
2	20%	0%	100% voltage & 100% current

Table 2 shows the results obtained from using binary classification on five test datasets. Tests 1, 2 and 4 were classified properly for both voltage and current regardless of the load applied. In test 3 and 5 with a 50% and 100% load respectively, the model classified the voltages as healthy correctly, but not for currents. The difference in classification between voltage and currents was expected as a loaded motor's currents differ from an unloaded motor's currents, and the algorithm was trained with data from no load conditions.

Table 2: Testing dataset results

Test	Fault	Loading	Testing Accuracy
1	40%	0%	100% voltage & 100% current
2	20%	100%	100% voltage & 100% current
3	0%	50%	100% voltage & 10% current
4	20%	50%	100% voltage & 100% current
5	0%	100%	100% voltage & 10% current

3.5. Confusion Matrices

Since both machine learning techniques obtained similar results, it also results in equal confusion matrices. The following shows the confusion matrixes obtained from running the binary classification and the PCA algorithm:

- Voltage and current training data sets:

$$C = \begin{bmatrix} 400 & 0 \\ 0 & 400 \end{bmatrix}, Classification\ accuracy = 100\%$$

- Voltage and current testing data sets:

$$C = \begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix}, Classification\ accuracy = 100\%$$

A classification accuracy of 100% means that the machine algorithm can properly distinguish between healthy and faulty data all the time. The confusion matrices and classification accuracies show that the results are as expected and the algorithms successfully classifies faults.

4. Conclusion & Future Work

Induction motors are increasingly being used, and with such use, eccentric motor faults will only increase. This paper looked at static eccentric faults only. Most papers relating to eccentric faults focused on frequency analysis and examined the harmonics present in the signal. In this paper, eccentric faults in induction motors are identified using machine learning techniques by using current and voltage data. The machine learning algorithms used are binary classification and principal component analysis (PCA). The binary classifier identified the eccentric faults correctly when using voltages regardless of the load applied while the classification failed when using current input from an induction motor with a load applied. The PCA algorithm classified the test data correctly for both voltage and current regardless of the load applied. The number of correct classifications is determined by using a confusion matrix and separate test datasets.

Future research will focus on identifying other eccentricity faults as well as on segregating the type of eccentricity faults in induction motors.

5. Acknowledgment

The authors would like to acknowledge the help provided by Dr. Wu-Sheng Lu of the University of Victoria for his course on machine learning optimization and NSERC for financial support.

6. References

[1] B. Plumer, N. Popovich and B. Migliozzi, "Electric Cars are Coming. How Long Unitl They Rule the Road?," The New York Times, 10 March 2021. [Online]. Available: https://www.nytimes.com/interactive/2021/03/10/climate/electric-vehicle-fleet-turnover.html. [Accessed 12 October 2021].

[2] J. Faiz and S. M. M. Moosavi, "Eccentricity Fault Detection - From Induction," Tehran, March 2016.

[3] M. Akar, S. Taskin, S. Seker and I. Cankaya, "Detection of Static Eccentricity for Permanent Magnet Synchronous Motors Using the Coherence Analysis," Turkish, Journal of Electrical Engineering and Computer Sciences, November 2010.

[4] W.-S. Lu, "ECE 403/503 Optimization for Machine Learning," University of Victoria, Victoria, 2021.

[5] N. Turk, "Fault Diagnosis Using Park's Vector Approach," International Research , Faridabad, 2016.

[6] I. T. Jolliffe and C. Jorge, "Principal component analysis: a review and recent developments," Philosophical Transactions of the Royal Society A, 2016.

[7] HandWiki, "Engineering: Alpha-beta transformation," HandWiki, [Online]. Available: https://handwiki.org/wiki/Engineering:Alpha%E2%80%93beta_transformation. [Accessed 18 May 2022].

[8] L. Smith, "A tutorial on Principal Components Analysis," 2002.