Classification of Induction Machine Faults using Time Frequency Representation and Particle Swarm Optimization

A. Medoued†, A. Lebaroud*, A. Laifa* and D. Sayad*

Abstract – This paper presents a new method of classification of the induction machine faults using Time Frequency Representation, Particle Swarm Optimization and artificial neural network. The essence of the feature extraction is to project from faulty machine to a low size signal time-frequency representation (TFR), which is deliberately designed for maximizing the separability between classes, a distinct TFR is designed for each class. The feature vectors size is optimized using Particle Swarm Optimization method (PSO). The classifier is designed using an artificial neural network. This method allows an accurate classification independently of load level. The introduction of the PSO in the classification procedure has given good results using the reduced size of the feature vectors obtained by the optimization process. These results are validated on a 5.5-kW induction motor test bench.

Keywords: Induction machine diagnosis, Ambiguity plane, Classification-optimal TFR, Time-frequency, Fisher’s discriminated ratio, Artificial neural network ANN, PSO.

1. Introduction

Today’s industry strives to improve performance and profitability while maintaining and improving safety. The challenges include reliability and safety operation of electric motors in an industrial process. Thus, very expensive scheduled maintenance is performed in order to detect machine problems before they may result in catastrophic failure [1-2]. Nowadays, maintenance cost reductions are the number one priority for electrical drive to prevent unscheduled downtimes and to increase operational effectiveness. Recent advances of signal processing techniques, such as artificial neural networks [3-8], wavelets [9], etc., have provided more powerful tools for fault diagnosis.

The problem of diagnosis systems is that they use signals either in the time or frequency domain. In our approach, instead of using a time or a frequency approach, it is potentially more informative to use both time and frequency. Time-frequency analysis of the motor current makes signal properties, related to fault detection, more evident in the transform domain [10].

Traditionally, the objective of time–frequency research is to create a function that will describe the energy density of a signal simultaneously in time and frequency. For explicit classification, it is not necessarily desirable to accurately represent the energy distribution of a signal in time and frequency. In fact, such a representation may conflict with the goal of classification, generating a TFR that maximizes the separability of TFRs from different classes. It may be advantageous to design TFRs that specifically highlight differences between classes [11-14].

Since all TFRs can be derived from the ambiguity plane, no a priori assumption is made about the smoothing required for accurate classification. Thus, the smoothing quadratic TFRs retain only the information that is essential for classification.

This classification allows us to proceed to an optimization routine based on particle swarm technique to find the appropriate size of the feature vectors in order to reduce calculation time and keep signal with relevant information within the vectors.

In this paper, we propose a classification algorithm based on the design of an optimized TFR from a time–frequency ambiguity plane in order to extract the feature vector. The optimal size of feature vectors is realized by the PSO algorithm. The PSO technique can generate high-quality solutions within shorter calculation time and stable convergence characteristic than any other stochastic methods [15-17].

Finally, a neural network-based decision criterion is used for classification. The goal of this work is the realization of an accurate classification system of motor faults such as bearing faults, stator faults, and broken bars rotor faults independently from the load level.

2. Classification Algorithm

The classification algorithm consists of the following three parts: extraction, optimization of features vectors and decision making. In the training stage, three optimal kernels are designed for separating four classes [18]:

† Corresponding Author: Dept. of Electrical Engineering, University of 20 Aout 1955-Skikda, Algeria. (amedoud@yahoo.fr)
* Dept. Dept. of Electrical Engineering, University of 20 Aout 1955-Skikda, Algeria. (lebaroud80@gmail.com)
Received: December 26, 2012; Accepted: August 3, 2013
A. Medoued, A. Lebaroud, A. Laifa and D. Sayad

1) Class of healthy motor;
2) Class of bearing fault;
3) Class of stator fault;
4) Class of broken bars.

The kernel design process selects, for each class, a number of locations from the time-frequency ambiguity plane. In the decision making stage, we propose an ANN classifier with the Levenberg Marquardt algorithm. The details of each step are described in the following sections.

3. Feature Extraction

3.1 Optimal TFR

For further details, we recommend the reader to review our previous works [19] and [20].

The expression of the TFR is given by:

\[ G[n,k] = \sum_{\eta=0}^{N-1} \sum_{\tau=0}^{N-1} \phi[\eta,\tau] A[\eta,\tau] e^{-j2\pi(n\eta + k\tau)} \]

(1)

The characteristic function for each TFR is \( A(\eta,\tau)\phi(\eta,\tau) \), \( \eta \) represents the discrete frequency shift and \( \tau \) represents the discrete time delay. This means that the optimal-classification representation \( TFR_i \) can be obtained by smoothing the ambiguity plane \( A(\eta,\tau) \) with an appropriate kernel \( \phi_{opt} \), which is an optimal classification kernel. The problem of designing the \( TFR_i \) becomes equivalent to designing the optimal classification kernel \( \phi_{opt}(\eta,\tau) \). This method, used to design kernels (and thus TFRs), optimizes the discrimination between predefined sets of classes.

Features can be extracted directly from \( A(\eta,\tau)\phi_{opt}(\eta,\tau) \) instead of the optimal classification TFR. This shortcut simplifies the computation complexity of the feature extraction by reducing the calculations.

3.2 Design of classification kernels

The kernel \( \phi_{opt}(\eta,\tau) \) is designed for each specific classification task. We determine \( N \) locations from the ambiguity plane, in such a way that the values in these locations are very similar for signals from the same class, but they vary significantly for signals from different classes. In our design, we use Fisher’s discriminant ratio, FDR [19-20], to get these \( N \) locations.

In our classification procedure, \( C-1 \) kernels must be designed for a \( C \)-class classification system. In order to avoid unnecessary computation to separate classes, we have proposed the principle of the remaining classes [11]. The discrimination between different classes is made by separating the class \( i \) from all the remaining classes \( \{i+1,\ldots,N\} \). In this case, the stator fault kernel is designed to discriminate the stator fault class from the other classes (rotor fault, bearing fault and healthy motor). The rotor fault kernel is designed to discriminate the rotor fault class from the remaining classes (bearing fault and healthy motor). The bearing fault kernel is designed to discriminate the bearing fault class from the healthy motor class. The advantage of the method lies in the optimum separation between the different classes.

4. Feature Vector Optimization

One objective of our approach is to minimize the signal size by the feature vector of a very small size without losing relevant information. Hence, the search for an optimum size of this vector provides a good compromise between the relevance of information and time consuming cost.

4.1 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), introduced by Eberhart and Kennedy [21], is based on the analogy of birds swarm and school of fish. In PSO, each individual called particle makes his decision using his own experience together with other individuals’ experience. In PSO, two different definitions are used: the individual best and the global best. As a particle moves through the search space, it compares its fitness value at the current position to the best fitness value it has ever attained previously. The best position that is associated with the best fitness encountered so far is called the individual best or pbest. The global best, or gbest, is the best position among all of the individual’s best positions achieved so far (Fig. 1).

Using the gbest and the pbest, the \( i \)th particle velocity is updated according to the following equation [22]:

\[ v_{i+1} = w v_i + c_1 r \times (pbest_i - s_i) + c_2 r \times (gbest - s_i) \]

(2)

Based on the updated velocities, each particle changes...
its position according to the equation:

\[ s_i^{k+1} = s_i^k + v_i^{k+1} \]  

(3)

Where \( w \) is a weighting function, \( c_j \) are acceleration factors and \( r_{and} \) is a random number between 0 and 1.

The following weighting function is usually utilized:

\[ w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter \]  

(4)

Where \( w_{max} \) is initial weight, \( w_{min} \) the final weight, \( iter_{max} \) is the maximum iteration number, and \( iter \) is the current iteration number.

The parameters used in this work are taken as follows [22-26]:

\( c_1 = c_2 = 2.05; w_{min} = 0.1; w_{max} = 0.9. \)

**Selection of maximum velocity:**

At each iteration step, the algorithm proceeds by adjusting the distance (velocity) that each particle moves in every dimension of the problem hyperspace. The velocity of the particle is a stochastic variable and is, therefore, subject to creating an uncontrolled trajectory, making the particle follow wider cycles in the problem space. In order to damp these oscillations, upper and lower limits can be defined for the velocity \( v_i \):

\[
\begin{align*}
\text{if } v_i \geq v_{\text{max}} & \text{ then } v_i = v_{\text{max}} \\
\text{elseif } v_i < -v_{\text{max}} \text{ then } v_i = -v_{\text{max}}
\end{align*}
\]

(5)

Most of the time, the value of \( v_{\text{max}} \) is selected empirically, according to the characteristics of the problem. It is important to note that if the value of this parameter is too large, then the particles may move erratically, going beyond a good solution; on the other hand, if \( v_{\text{max}} \) is too small, then the particle’s movement is limited and the optimal solution may not be reached.

Fan and Shi [27] have shown that an appropriate dynamically changing \( v_{\text{max}} \) can improve the PSO algorithm performance. To ensure a uniform velocity we fixed \( v_{\text{max}} \) according to many run tests.

**Integer PSO formulation:**

In the case where integer variables are included in the optimization problem such as a size of feature vector, the PSO algorithm can be reformulated by rounding off the particle’s position to the nearest integer. Mathematically, (3) and (4) are still valid, but once the new particle’s position is determined in the real-number space, the conversion to the integer number space must be done.

**4.2 Fitness function**

For searching an optimized size of the feature vector based on PSO algorithm, a fitness function is needed. In this work, we consider the variance calculated for every size of the feature vector as the fitness for this size and the goal is to optimize this fitness.

**5. Classification Using Neural Networks**

In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. The learning procedure tries to find a set of connections \( w \) that gives a mapping that fits well the training set.

Furthermore, neural networks can be viewed as highly nonlinear functions with the basic form:

\[ F(x, w) = y \]  

(6)

Where \( x \) is the input vector presented to the network, \( w \) are the weights of the network, and \( y \) is the corresponding output vector approximated or predicted by the network. The weight vector \( w \) is commonly ordered first by layer, then by neurons, and finally by the weights of each neuron plus its bias.

This view of network as a parameterized function will be the basis for applying standard function optimization methods to solve the problem of neural network training.

**5.1 Network training as a function optimization problem**

As mentioned previously, neural networks can be viewed as highly non-linear functions. From this perspective, the training problem can be considered as a general function optimization problem, with the adjustable parameters being the weights and biases of the network, and the Levenberg-Marquardt can be straightforward applied in this case.

**5.2 Levenberg-marquardt algorithm**

Basically, it consists in solving the equation:

\[ (J^T J + \lambda I)\delta = J^T E \]  

(7)

Where \( J \) is the Jacobian matrix (Eq. 8), \( \lambda \) the Levenberg's damping factor, \( \delta \) the desired updated weight vector and \( E \) the error vector containing the output errors for each input vector used on training the network. The \( \delta \) tells us by how much we should change our network weights to achieve a (possibly) better solution.
The $J^T J$ matrix may also be known as the approximated Hessian. The $\lambda$ damping factor is adjusted at each iteration, and guides the optimization process. If the reduction of $E$ is rapid, a smaller value can be used, bringing the algorithm closer to the Gauss-Newton algorithm, whereas if an iteration gives insufficient reduction in the residual, $\lambda$ can be increased, giving a step closer to the gradient descent direction.

### 5.3 Computing the Jacobian

An $N$-by-$M$ matrix of all first-order partial derivatives of a vector-valued function. $N$ is the number of entries in our training set and $M$ is the total number of parameters (weights + biases) of our network. It can be created by taking the partial derivatives of each output in respect to each weight, and has the form

$$
J = \begin{bmatrix}
\frac{\partial F(x_1, w)}{\partial w_1} & \ldots & \frac{\partial F(x_1, w)}{\partial w_M} \\
\vdots & \ddots & \vdots \\
\frac{\partial F(x_n, w)}{\partial w_1} & \ldots & \frac{\partial F(x_n, w)}{\partial w_M}
\end{bmatrix}
$$

(8)

Where $F(x_i, w)$ is the network function evaluated for the $i$th input vector of the training set using the weight vector $w$ and $w_j$ is the $j$th element of the weight vector $w$ of the network.

### 5.4 General Levenberg-Marquardt algorithm

As stated earlier, the Levenberg-Marquardt consists basically in solving (11) with different values of $\lambda$ until the sum of squared error decreases. So, each learning iteration (epoch) will consist of the following basic steps:

1. Compute the Jacobian
2. Compute the error gradient: $g = J^T E$
4. Solve $(H + \lambda I) \delta = g$ to find $\delta$
5. Update the network weights $\theta$ using $\delta$
6. Recalculate the sum of squared errors
7. If the sum of squared errors has not been decreased, discard the new weights, increase $\lambda$ using $v$ and go to step 6.
8. Else decrease $\lambda$ using $v$ and stop.

Variations of the algorithm may include different values of $v$, one for decreasing $\lambda$ and another for increasing it. Others may solve $(H + \lambda \text{diag}(H)) \delta = g$ instead of $(H + \lambda I) \delta = g$, while others may select the initial $\lambda$ according to the size of the elements on $H$, by setting $\lambda_0 = t \max(\text{diag}(H))$, where $t$ is a chosen value.

We can see that we will have a problem if the error does not decrease after some iteration. In this case, the algorithm also stops if $\lambda$ becomes too large [28-29].

### 6. Experiment Results

The experimental data are collected in Ampère Laboratory, University of Lyon. The experimental bench consists of a three-phase asynchronous-motor squirrel cage Leroy Somer LS 132S, IP 55, Class F, T °C standard = 40 °C. The motor is loaded by a powder brake. Its maximum torque (100 Nm) is reached at rated speed.

This brake is sized to dissipate a maximum power of 5kW. Fig. 2 shows the motor bench. The wear obtained on the bearings is a real one (Fig. 3). For the rotor fault, the bar has been broken by drilling the bar of the squirrel cage (Fig. 4). For simulating the fault of imbalance stator, imbalanced power is obtained with a variable autotransformer placed on a phase of the network (Fig. 2).

The acquisition system used to measure these signals consists of eight differential inputs used to measure currents sampled up to 20 MHz 14-bit.

The current signals sampling rate is 20 kHz. The number of samples per signal rises to $N=100000$ samples on an acquisition period of 5s. The data acquisition set consists of 15 examples of stator current recorded on different levels of load (0%, 25%, 50%, 75% and 100%). Different operating conditions for the machine were considered, namely, healthy, bearing fault, stator fault and rotor fault. The training set is carried out on first ten current examples. The last five current examples are used to test the classification.
Each signal is passed through a lowpass filter and resampled with a downsampling rate of 50. Only the range of the required frequencies is preserved. The lowpass filter is used in order to avoid aliasing during downsampling. The dimension of ambiguity plane is $200 \times 200 = 40000$ points; by considering symmetry compared to the origin, we retain only the quarter of ambiguity plane, which corresponds to $N=10000$. We designed three kernels: stator fault kernel, rotor fault kernel and bearing fault kernel [18]. Fisher’s point locations in the Doppler-delay plane are ranged in the feature vectors $\{FV_1, \ldots, FV_N\}$ as training database of the neural network. In neural network, if there are too few neurons in the hidden layer, the network may not contain sufficient degrees of freedom to form a representation. If too many neurons are defined, the network might become overtrained. Therefore, an optimum design of the neurons number is required. In this work, we used one hidden layer with a number of different neurons to determine the suitable network. As a stop criterion we intended a goal of $10^{-12}$ which defines the convergence of the algorithm. The goal is reached in a minimum number of epochs 16 and 24, Fig. 5 and 6 respectively.

The training algorithm gives a better performance for a number of 5 neurons in the hidden layers for the three kernels (Table 1).

Table 1. Misclassification results

<table>
<thead>
<tr>
<th>Neurons number</th>
<th>Actual error</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=3</td>
<td>FDR1 1/15</td>
</tr>
<tr>
<td>N=4</td>
<td>FDR1 1/15</td>
</tr>
<tr>
<td>N=5</td>
<td>FDR1 1/15</td>
</tr>
<tr>
<td>N=6</td>
<td>FDR1 1/15</td>
</tr>
</tbody>
</table>

Fig. 7 shows that for 15 test vectors, in case of Kernel 1, 14 were classified which indicates that the classification error is acceptable. This is also true for the two other kernels. Furthermore, the increase of the size of feature vector reduces significantly this error. However, the classification error is minimized when we increase the number of training vectors to 35 vectors (10 vectors of stator currents at 0% of charge, 5 at 25%, 5 at 50%, 5 at 75% and 10 at 100% of rated charge). Fig. 8 shows clearly a marked improvement in the classification process.

The objective of introducing the PSO is the optimization.
We have introduced the PSO algorithm to optimize the size of the feature vectors. Our classification is based on the ambiguity Doppler-delay plane where all the TFRs can be derived by a suitable choice of a kernel. Each type of fault was characterized by a specific kernel. The classification algorithm was tested by comparison with experimental data collected from the stator current measurement at different load levels. The assignment of signal was made by an ANN classifier. The results show that the new algorithm, with the neural network classifier as a decision criterion and the PSO as an optimizing technique, is able to detect and diagnose faults with acceptable accuracy and time consuming calculations compared to the case without PSO optimisation, independently of the load condition and the fault type.

References


Ammar Medoued He received the degree of Doctor of Sciences from University of Skikda, Algeria in Electrical Engineering. He is currently a Lecturer at the University of Skikda and the Head of the Department of Electrical Engineering. His main research field is Electrical Machine Diagnosis.

Abdesselam Lebaroud was born in Constantine, Algeria, in 1969. He received the PhD degree in electrical engineering from University Claude Bernard Lyon I, Ampere laboratory, France, in 2007. Currently, he is a Professor at the Department of Electrical Engineering, University of Skikda. He carried out researches on diagnosis of electrical machines at LGEC of Constantine.
Abdelaziz LAIFA He worked in oil industrial field for many years. Since 2001, he has been with the University of Skikda as a lecturer and researcher, where he received the PhD in Electrical Power Engineering in 2012. His main interests are Power Systems Analysis and Control using intelligent programming and meta-heuristic methods.

Djamal Sayad He received the degree of Magister in Electronics from the University of Constantine 1998. He is actually a Lecturer at the University of Skikda, Algeria. His main field of research is Signal Processing, Diagnosis.