Chapter Three

THEORETICAL FRAMEWORK

Chapter 3 provides technical discussions related to the study. The technical concepts behind PMSMs, Field-Oriented Control, Neural Networks, and Genetic Algorithms are discussed. Indicated here are the relevant equations, concepts, and algorithms used for the study shown by means of equations and figures.

3.1 Sensorless Field-Oriented Control of PMSM

This section will discuss the relevant theoretical information in the sensorless implementation of the field-oriented control for a PMSM.

3.1.1 Permanent Magnet Synchronous Motor

This research will make use of Permanent-Magnet Synchronous Motors (PMSMs) as the induction motor as they are known for being more efficient than other motors and having a higher torque density in terms of torque per unit volume [2].

A PMSM's rotor consists of surface-mounted permanent magnets and its stator has a Y-connected wind located every 120º. The three-phase PMSM is regarded as a nonlinear time-varying system that can be simplified into a two-phase voltage model in synchronous rotating d-q coordinates via the concepts of Clark and Park transformations.

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \begin{bmatrix} R_s + \frac{d}{dt}L_q & -\omega_e L_q \\ \omega_e L_q & R_s + \frac{d}{dt}L_d \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_e \lambda_f \end{bmatrix} \quad (3.1)$$

where u_d, u_q represents the dq-axis stator voltages of the PMSM; i_d, i_q, representing the dq-axis stator currents; L_d, L_q , representing the dq-axis stator inductances; R_s, represents the stator winding resistance of the PMSM; ω_e, represents the rotor's angular speed; and λ_f, represents the permanent magnets' flux-linkage.

As mentioned, the d-q coordinate model can be transformed into an equivalent model in a stationary α-β coordinate system. This is achieved by using the inverse Park transform on the d-q model.

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} R_s + \frac{d}{dt}L_s & 0 \\ 0 & R_s + \frac{d}{dt}L_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} \quad (3.2)$$

where u_α, u_β represents the alpha and beta stator voltages; i_α, i_β represents the alpha beta stator currents; R_s represents the stator winding resistance of the PMSM; L_s represents the stator winding inductance; and e_α, e_β represents the alpha beta equivalent back-EMF voltages.

3.1.2 Sensorless Field-Oriented Control Operation

Field oriented control (FOC) or vector control of permanent magnet synchronous motors is a motor drive control scheme or approach that improves the performance of an induction motor driven by an inverter. This approach is able to obtain near-instantaneous response in torque by transferring from a steady-state condition to another.

In FOC discussions, spatial quantities such as the magneto-motive force (mmf) and the rotor flux are taken with great significance as they allow simplicity in the modelling and understanding of the motor control scheme by being representative of the individual phase currents. Figures 3.1 and 3.2 represents the sensorless field-oriented control operation in terms of a top-level overview and a more detailed block diagram.

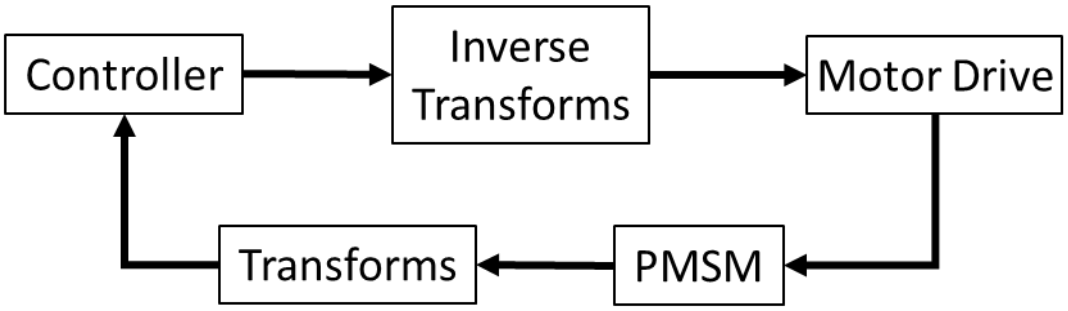


Figure 3.1 Main Control Loop of Field Oriented Control Scheme

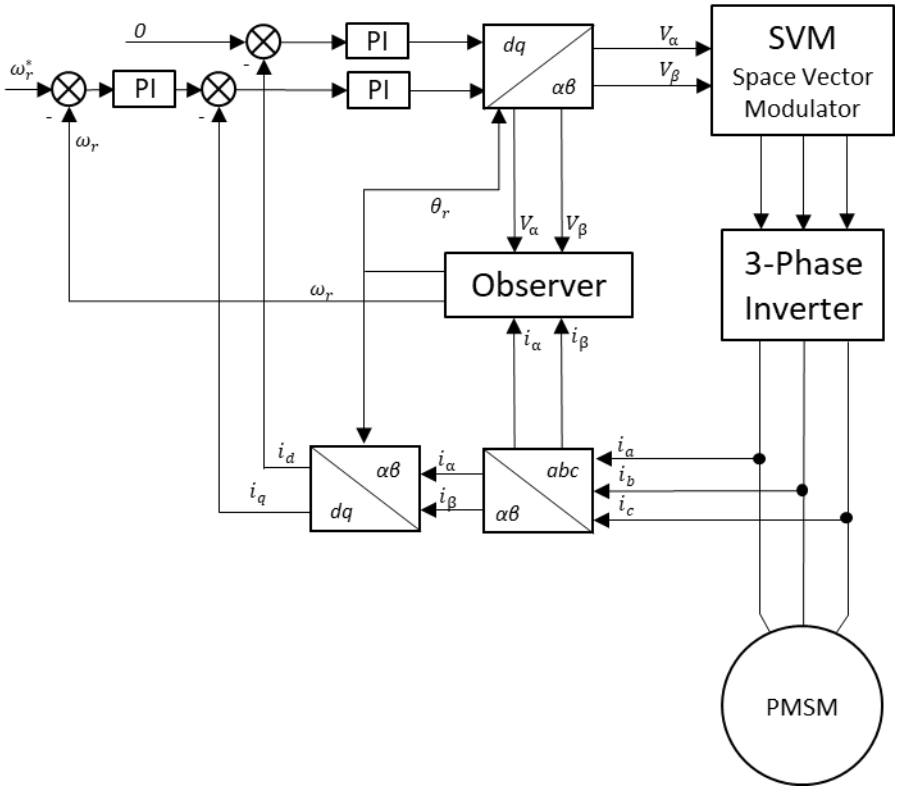


Figure 3.2 Sensorless Field Oriented Control Block Diagram

3.1.3 Motor Phase Currents Measurement

As the motor's phase currents serve as the feedback for the FOC scheme, current measurements are required. In actual applications, the analog signals of the values of the motor's phase currents need to be fed into an ADC port of the controller or to a standalone ADC.

Since isolation between the motor current and the feedback loop is not required, a simple shunt resistor can be implemented for each of the phase currents. In simulation, the presence of shunt resistors may be disregarded.

3.1.4 Clarke and Park Transforms

One of the main features of the FOC scheme is the use of transforms to generate the equivalent flux and torque of an induction motor from various measured and/or estimated parameters. These concepts are applicable due to the characteristic of AC motors for analyzing three-phase parameters in terms of complex space vectors. The transformations are used to deliver simplification of analysis when working with three-phase vectors.

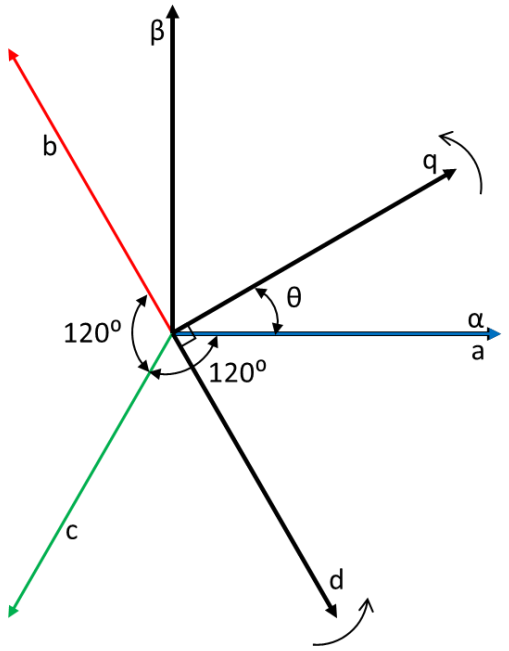
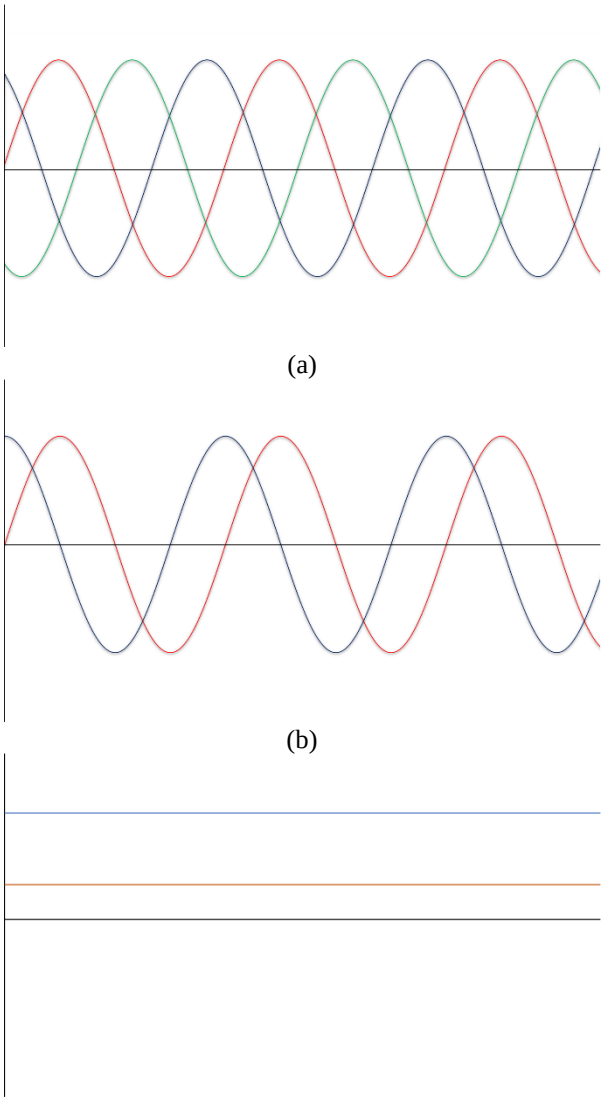


Figure 3.3 Representation of Clarke and Park transforms' equivalent vectors

Clarke Transform

With the motor's phase currents serving as the inputs, the first transformation is the alpha-beta transformation or also called as the Clarke Transformation wherein the three-phase system is projected into a two-dimension orthogonal system. The mathematics behind the Clarke transform isolates the common-mode component among the three vectors.



(a)

(b)

(c)

Figure 3.4 Motor Current Representations (a) 3-Phase Motor Currents (2) 2-D Orthogonal System (3) Rotating 2-D Orthogonal System

Figure 3.4 (a) shows the three-phase sinusoidal motor currents, i_a, i_b, i_c , while Figure 3.4 (b) is the resultant alpha-beta axis projection via a Clarke transformation. The formulas used for the Clarke transform and the inverse Clarke transform used in this study are shown in Equations 3.3 and 3.4, respectively.

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \quad (3.3)$$

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \frac{3}{2}\begin{bmatrix} \frac{2}{3} & 0 \\ -\frac{1}{3} & \frac{\sqrt{3}}{3} \\ -\frac{1}{3} & -\frac{\sqrt{3}}{3} \end{bmatrix}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \quad (3.4)$$

where i_a, i_b, i_c represents the three-phase motor currents; i_α, i_β represents the alpha and beta axis projections of the motor currents.

Park Transform

The Park transform will deliver the 2-D orthogonal system into a rotating 2-D orthogonal system. This will deliver almost DC values to be used as feedback values for the control loop. This transform enables the ease of control by manipulating these DC values instead of sinusoidal ones. The formulas used for Park Transform are given by equations 3.5 and 3.6.

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \sin\theta & -\cos\theta \\ \cos\theta & \sin\theta \end{bmatrix}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \quad (3.5)$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} i_d \\ i_q \end{bmatrix} \quad (3.6)$$

3.1.5 Flux, Speed, and Torque Controllers

Proportional-Integral Controllers, similar to Figure 3.5, are used in the proposed control system to reduce certain error values to zero.

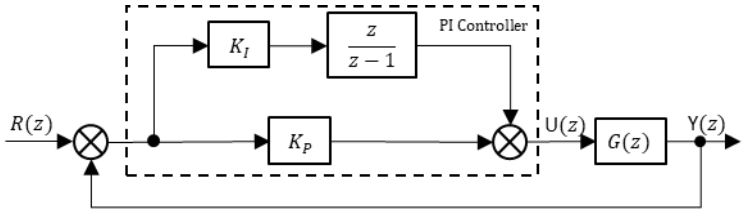


Figure 3.5 Proportional-Integral Controller

Reference values for flux, speed, and torque are introduced. Reference flux should be zero since this follows the concept of FOC. The reference speed is a user input. The reference torque value is simply obtained from the output of the PI controller for speed.

3.1.6 Space Vector Modulation

Space Vector Modulation (SVM) is the control method to be used in FOC in driving the 3-phase PMSM in this proposal. The SVM is utilized due to its ability to make instantaneous changes in its output phasor's angular position which is advantageous when used for field-oriented control or vector control of a 3-phase motor.

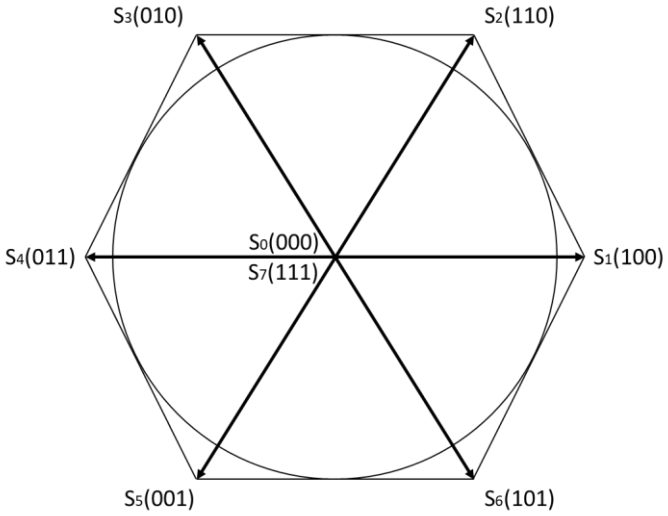


Figure 3.6 Space Vector Modulation States

In figure 3.6, there are eight possible states for the switching power devices, wherein we have 3 pairs of switches. Each state represents a combination of which switch is on and which is off that determines the output phasor. State 0 is where all primary switches are off and all secondary switches are on, while State 7 is the opposite. This method of controlling the output phasor that rotates at a constant rate in order to produce a corresponding set of balanced 3-phase voltages.

3.1.7 Sliding Mode Observer for Position and Speed Estimation

In this study, the method used to estimate the rotor position and speed is the usage of the back EMF-based sliding mode observer. In the proposed sensorless control of a PMSM, the hardware mechanical sensors for speed and position are replaced with an estimator. The model-based estimator to serve as reference in this study is shown in Figure 3.7.

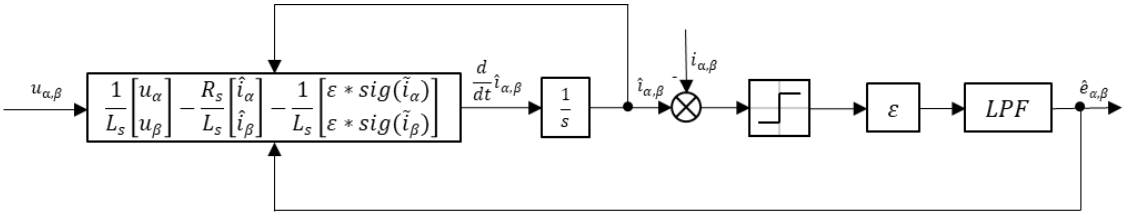


Figure 3.7 Sliding Mode Observer Block Diagram

Using the equation used in characterizing the motor in Eq. 3.7, the proposed back-EMF SMO is based on the following equation:

$$\frac{d}{dt}\begin{bmatrix}\hat{\imath}_\alpha\\\hat{\imath}_\beta\end{bmatrix} = \frac{1}{L_s}\begin{bmatrix}u_\alpha\\u_\beta\end{bmatrix} - \frac{R_s}{L_s}\begin{bmatrix}\hat{\imath}_\alpha\\\hat{\imath}_\beta\end{bmatrix} - \frac{1}{L_s}\begin{bmatrix}\varepsilon * sig(\tilde{\imath}_\alpha)\\\varepsilon * sig(\tilde{\imath}_\beta)\end{bmatrix} \quad (3.7)$$

where $\hat{\imath}_\alpha, \hat{\imath}_\beta$ represents the *estimated* alpha and beta equivalent stator currents; $\tilde{\imath}_\alpha, \tilde{\imath}_\beta$ represents the estimation *error* as shown in the next equation:

$$\boldsymbol{s} = \begin{bmatrix}s_\alpha\\s_\beta\end{bmatrix} = \begin{bmatrix}\tilde{\imath}_\alpha\\\tilde{\imath}_\beta\end{bmatrix} = \begin{bmatrix}\hat{\imath}_\alpha - i_\alpha\\\hat{\imath}_\beta - i_\beta\end{bmatrix} \quad (3.8)$$

where **s** is the sliding mode surface of the proposed back-EMF SMO and is based on the error between the estimated alpha and beta currents against the actual measurements that are transformed from the current shunt measurements.

The proposed SMO will use the sigmoid function *sig* as its switching function. The sig function is stated as:

$$sig\begin{pmatrix}\tilde{\imath}_\alpha \\ \tilde{\imath}_\beta\end{pmatrix} = \frac{2}{1+e^{[-\sigma(\hat{\imath}_s - i_s)]}} - 1 \qquad (3.9)$$

where $\hat{\imath}_s = [\hat{\imath}_\alpha \quad \hat{\imath}_\beta]^T$ representing the *estimated* alpha and beta stator currents; $i_s = [i_\alpha \quad i_\beta]^T$ representing the transformed measured stator currents; and σ is a positive real number.

The stability of the conventional SMO is determined by a Lyapunov function:

$$V = \frac{1}{2}(\boldsymbol{s}^T\boldsymbol{s}) = \frac{1}{2}(s_\alpha^2 + s_\beta^2) \qquad (3.10)$$

$$\frac{d}{dt}\begin{bmatrix}s_\alpha \\ s_\beta\end{bmatrix} = -\frac{R_s}{L_s}\begin{bmatrix}s_\alpha \\ s_\beta\end{bmatrix} + \frac{1}{L_s}\begin{bmatrix}e_\alpha \\ e_\beta\end{bmatrix} - \frac{\varepsilon}{L_s}\begin{bmatrix}sig(s_\alpha) \\ sig(s_\beta)\end{bmatrix} \qquad (3.11)$$

According to the Lyapunov stability theorem, the system is considered to be stable when $\dot{V} < 0$ for V > 0. The equation for the estimation error can be obtained by subtracting the state equation of the stator currents from Eq. 3.7, as shown in Eq. 3.11. Differentiating Eq. 3.10 and consequently substituting into Eq. 3.11 yields:

$$\dot{V} = -R_s(s_\alpha^2 + s_\beta^2) - \varepsilon[s_\alpha sig(s_\alpha) + s_\beta sig(s_\beta) + (s_\alpha e_\alpha + s_\beta e_\beta) < 0 \qquad (3.12)$$

$$\dot{V} = [s_\alpha e_\alpha - \varepsilon s_\alpha sig(s_\alpha)] + [s_\beta e_\beta - \varepsilon s_\beta sig(s_\beta)] - R_s(s_\alpha^2 + s_\beta^2) < 0 \qquad (3.13)$$

Given that $-\frac{R_s}{L_s}s_\alpha^2 < 0$ and that $-\frac{R_s}{L_s}s_\beta^2 < 0$, the SMO gain should be $\varepsilon > \max(|e_\alpha|, |e_\beta|)$. Based on that, the observer gain ε should be relatively large to ensure

the stability of the SMO system. The values for the motor's alpha and beta equivalent back-EMF is represented by:

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = \begin{bmatrix} \varepsilon * sig(\tilde{\imath}_\alpha) \\ \varepsilon * sig(\tilde{\imath}_\beta) \end{bmatrix} = \lambda_f \omega_e \begin{bmatrix} -\sin\theta_e \\ \cos\theta_e \end{bmatrix} \quad (3.14)$$

where λ_f is the flux linkage of the permanent magnets; ω_e is the rotor's angular speed. Once the back-EMF is obtained, the estimated values for the rotor position and speed can be obtained by the following equations:

$$\tilde{\theta} = -\operatorname{atan}^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) \quad (3.15)$$

Where $\hat{\theta}$ is the estimated rotor position, $\hat{e}_\alpha$ and $\hat{e}_\beta$ are the estimated alpha-beta back-EMF values. To execute the rotor speed estimation, a PLL estimator is used. The PLL's input is θ_{err} and is explained by the following equations:

$$\hat{\theta} - \tilde{\theta} \approx \sin\left(\tilde{\theta} - \hat{\theta}\right) = \sin\left(\tilde{\theta}\right) \cdot \cos\left(\hat{\theta}\right) - \cos(\tilde{\theta}) \cdot \sin(\hat{\theta}) \quad (3.16)$$

$$\theta_{err} = \hat{\theta} - \tilde{\theta} \quad (3.17)$$

Where $\tilde{\theta}$ is the position value determined from equation 3.15, $\hat{\theta}$ is the position obtained from integrating the rotor speed in the PLL, and θ_{err} is the position error that serves as the input of the phase-locked loop as shown in Figure 3.8.

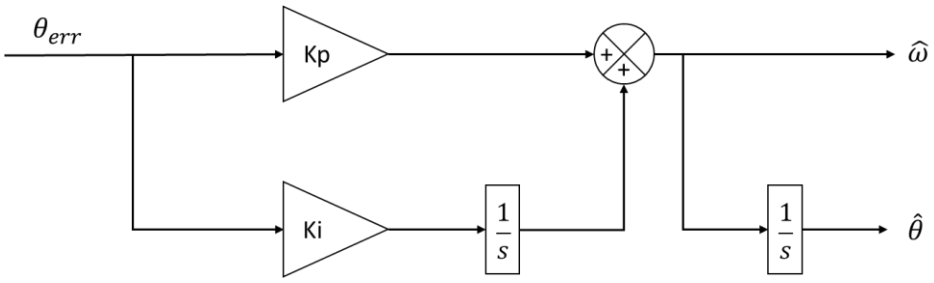


Figure 3.8 Phase Locked Loop Block Diagram

In figure 3.8, the position error θ_{err} is fed into a PI controller, the output of the PI controller is the estimated rotor speed $\hat{\omega}$. Integrating further, the estimated position $\tilde{\theta}$ is

obtained. The scheme, when using sine and cosine values highly reduces spikes from $-\pi \; to \; \pi$ [78].

3.2 Artificial Neural Networks

This section will discuss relevant theoretical discussions that are pertinent to the design and implementation of the proposed neural network estimator.

3.2.1 Feedforward Neural Networks

Feedforward neural networks are artificial neural networks that involves a single flow or direction for the network's information, hence the term feedforward. In this type of neural network, there is no cycle or cyclical pattern present within the nodes. A single node in a neural network can be represented by Figure 3.9.

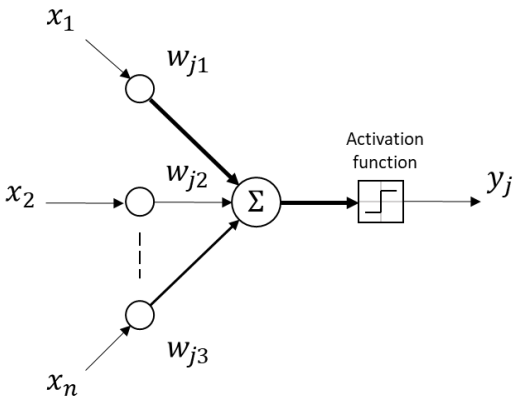


Figure 3.9 Single Neuron Representation

The figure above is a representation of a node j with inputs $x_1, x_2, \dots, x_n$. Each input has a corresponding weight i.e., w_{j1}, which also serves as its multiplier. The weighted inputs are then summed up and fed into an activation function. Basically, entire neural network architectures are made up of these nodes or *neurons.* A common representation of a feedforward neural network is shown in Figure 3.10.

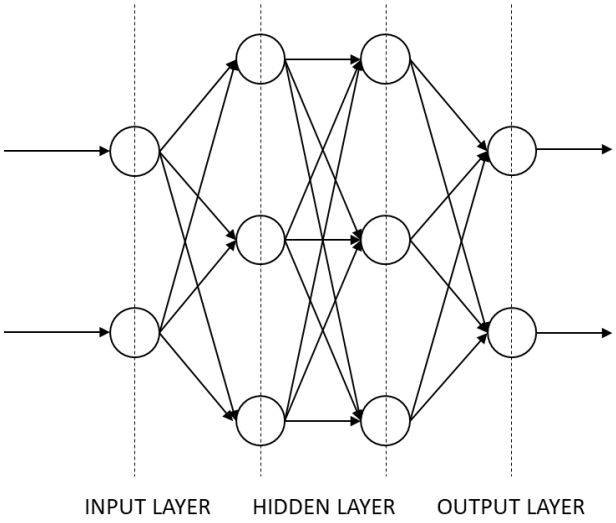


Figure 3.10 Feed-forward Neural Network

In a feedforward neural network, the input layer consists simply of the input parameters or information of the network, it is then feed to a hidden layer of neuron/s. There is a variable number of hidden layers in every architecture fitting the needs of the system, should there be more than one hidden layer, it will be referred to as a multi-layer feedforward neural network. The nodes that are present in the hidden layer performs the neural network's processing of the information [79]. The output layer will serve as the final layer for receiving the information already processed by the hidden layer.

3.2.2 Recurrent Neural Networks

Basing off feedforward neural networks, recurrent neural networks (RNNs) have introduced an additional feature. Such features are attractor dynamics and the ability to store information for later use, as well as its ability to handle inputs or outputs that are time-varying with the use of its inherent temporal operation [80].

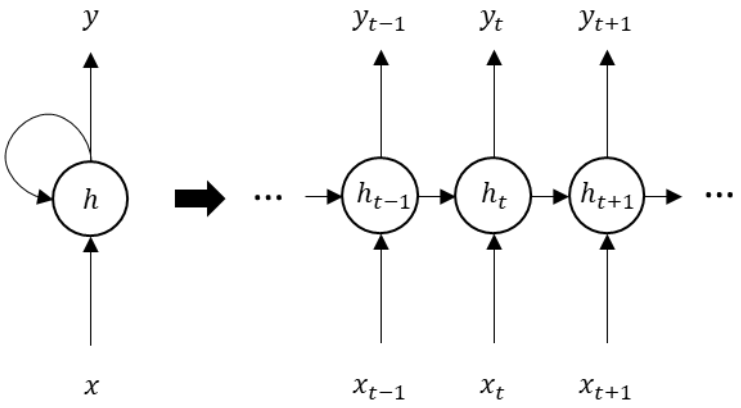


Figure 3.11 Recurrent Neural Network Neurons

In an RNN, the previous output also serves as the input for the current calculations being made in a single neuron. The previous output serves as a form of feedback as shown in Figure 3.11.

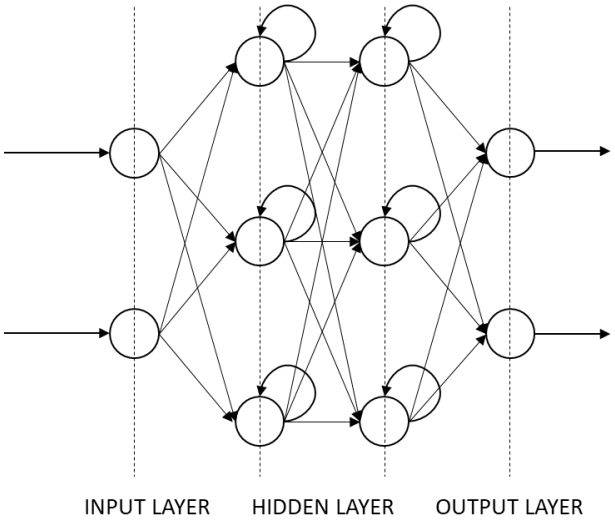


Figure 3.12 Recurrent Neural Network

Figure 3.12 shows a similar neural network architecture to that of the feedforward neural network with the difference of using RNN neurons in the hidden layer.

3.2.3 Training Algorithms

In order for neural networks to *learn*, training algorithms are used to manipulate or adjust the weights and biases of the network according to a loss function. Considering the complexity of a network, these algorithms exhibit different approaches in order for a network to reach its local minima each with its own set of advantages and disadvantages. Multiple learning or training algorithms have already been used in neural estimators as shown in Table 2.2.

One of the most popular training algorithms is the back-propagation algorithm as popularized by [81]. In back-propagation, the basic concept is the efficient computation of gradients by means of propagation from the output to the input of the network [82]. From the computation of the gradients, the effect of each weight and bias of the prior layer on the desired output is considered and will be used to determine the change in weight value when all effects are considered. This process in then continued on the preceding layers until the input before propagating forward and observing the new errors.

Other popular methods used as training algorithms are also considered in this paper. Such as the Gradient Descent Method, Gauss-Newton Method, and the Levenberg-Marquardt Method.

The *Gradient Descent Method* is considered as a general minimization algorithm which should update the parameters in a general *downhill* direction – towards the local minima. This method has good convergence in simpler function problems [83]. Gradient descent's parameter update for determining the direction of the steepest descent to the local minima is given by Eq 3.18.

$$h_{gd} = \alpha \boldsymbol{J}^T \boldsymbol{W}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \qquad (3.18)$$

where $\boldsymbol{J}$ represents the *m x n* Jacobian matrix that represents the local sensitivity of the function $\hat{\boldsymbol{y}}$ to the variation in parameters, $\boldsymbol{W}$ is the weighting matrix, $\boldsymbol{y}$ is the data, $\hat{\boldsymbol{y}}$ is the curve-fit function, and α is a positive scalar that determines the step length towards the determined direction.

The *Gauss-Newton Method* is more appropriate for moderately-sized problems as it typically converges faster than the gradient descent method [84]. The normal equation in a Gauss-Newton method is given by Eq 3.19

$$[\boldsymbol{J}^T\boldsymbol{W}\boldsymbol{J}]h_{gn} = \boldsymbol{J}^T\boldsymbol{W}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \quad (3.19)$$

The *Levenberg-Marquardt Method* is simple an adaptive algorithm utilizing both the gradient descent and Gauss-Newton methods and is given by Eq 3.20.

$$[\boldsymbol{J}^T\boldsymbol{W}\boldsymbol{J} + \lambda\, diag(\boldsymbol{J}^T\boldsymbol{W}\boldsymbol{J})]h_{gn} = \boldsymbol{J}^T\boldsymbol{W}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \quad (3.20)$$

where λ is a damping parameter initialized at a large value so that initial updates are taken as small steps in the steepest-descent direction. When the value of λ is small, the update leans towards a Gauss-Newton update and a large value will result to a gradient descent update.

3.3 Sensorless FOC Neural Network Model

In this paper, the sliding-mode observer of a sensorless FOC model will be replaced with a neural network. Using Figure 3.2 as a reference, the estimator block used is a model-based estimator that uses a sliding-mode observer to generate an estimated value for the back-EMF. The back-EMF is then used to calculate for the position and speed of the motor. In this paper, the entire SMO block will be replaced by an artificial neural network as shown in Figure 3.13.

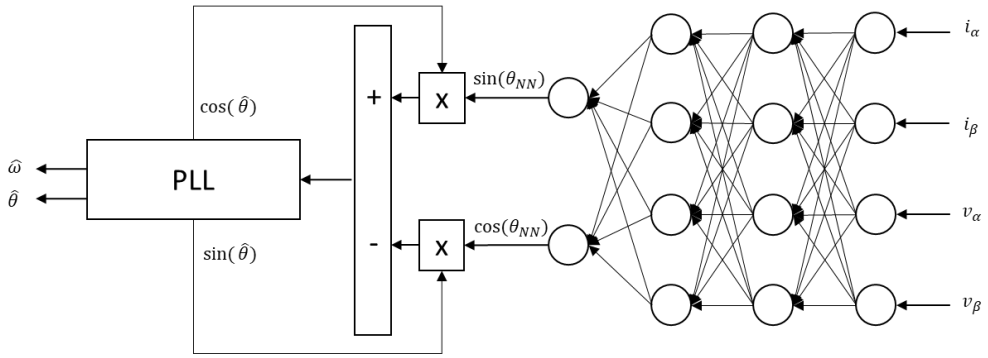


Figure 3.13 Neural Network-based Estimator

In this NN-based estimator, the inputs are set as the alpha-beta currents and voltages of the PMSM, same as the back-EMF based estimator. The output of the neural network comes in the form of $\sin(\theta_{NN})$ and $\cos(\theta_{NN})$ that also represent, as shown in Eq 3.14, the alpha-beta back-EMF of the motor.

In securing the dataset for this NN-based estimator, the alpha-beta currents and voltages, and the actual rotor position is obtained through simulation, the latter is the target value. Instead of estimating the actual position θ, the equivalent sine and cosine position is estimated to avoid the modulo 2π effect.

3.4 Genetic Algorithm for Hyperparameter Optimization

To determine the optimized hyperparameters for the artificial neural network, genetic algorithms (GA) is to be implemented. Genetic algorithms are one of the popular optimization algorithms that are biologically based. As first introduced in [85], GA optimizes the parameters based on the *fitness* of an individual.

The fitness of an individual is determined by a fitness function, in this study the fitness function would be based on the neural network's accuracy or error rate. An individual is considered as the solution wherein in this study would refer to the set of hyperparameters of the neural network. A set of individuals is considered as a population. Here, the optimization process based on GA that is used in this study will be discussed.

A	1	1	0	0	1	1	0	0
B	1	0	1	0	1	0	1	0
C	1	1	1	1	0	0	0	0

Figure 3.14 Representation of Genes in Genetic Algorithm

The initial population in this GA is generated randomly or heuristically. Then, the fitness of each individual is evaluated. The selection of parents for the reproduction of the next generation is determined by various methods and are based on the individual fitness score. If the fitness proportionate selection method is used as the genetic operator

or referred to as the *selection operator*, the probability of a certain individual to be selected is given by Eq 3.21.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \quad (3.21)$$

where f_i is the fitness score of an individual i, and N is the number of individuals in the population. This would allow the algorithm to pass along the better genes to the next generation. The total probability for a generation should be approximately equal to 1.

After determining the fitness score for each individual and determining the pairings for reproduction, an operator is then used to generate the offspring from the parents. The c*rossover operator* takes genes from both parents to create the genes for the child individual. Different methods of performing the crossover operator includes the edge recombination operator, cut and splice crossover method, and the uniform crossover method. Figure 3.15 shows the offspring of individuals A and B referred to as A'. The crossover method in this case crossed the genes every two bits. The child basically bears a combination of genes from the parents.

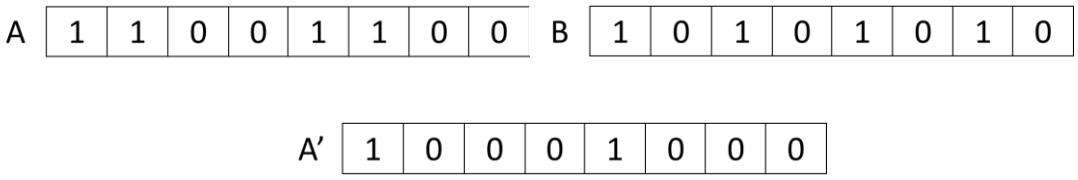


Figure 3.15 Representation of Crossover Operation

There also exists the *mutation operator* to address the issue of convergence to a local minimum by diversifying the genes. Like the selection and crossover operators, the mutation operator is selected based on the solution representation in the individuals given the design of the solution space.

The algorithm terminates when an individual meets the criteria for an acceptable solution or when a maximum number of generations is reached. In a neural network application, this criterion may be represented by the network's accuracy.

3.5 Conceptual Framework

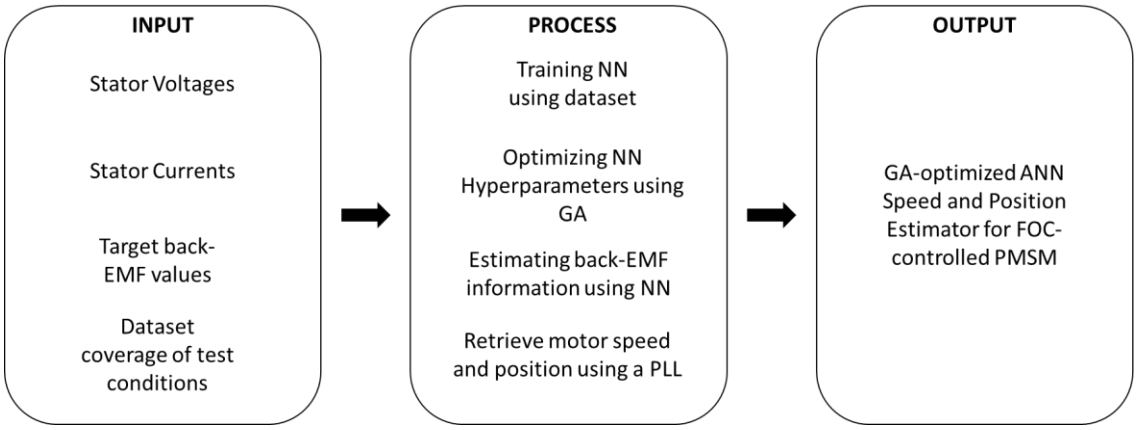


Figure 3.16 Research Paradigm

The research paradigm of this study, as shown in Figure 3.16, highlights the independent variables that will be used to meet the study's objectives. The process of training and optimizing the neural network-based observer starts with the data information in the form of the stator voltages and currents, which also serves as the inputs of the neural network itself, the target back-emf values which is the target variables when training the neural network, all falling under the dataset taken over all valid test conditions.

With the dataset, the NN can be trained and have its hyperparameters optimized by the implementation of a genetic algorithm. The trained and optimized NN can estimate the back-EMF using the information on the stator voltages and currents. The PLL will retrieve the speed and position estimated by the NN.